

# B I O I N F O R M A T I C S

**Kristel Van Steen, PhD<sup>2</sup>**

**Montefiore Institute - Systems and Modeling**

**GIGA - Bioinformatics**

**ULg**

**[kristel.vansteen@ulg.ac.be](mailto:kristel.vansteen@ulg.ac.be)**

## **CHAPTER 5: SEQUENCE COMPARISON**

### **1 The biological problem**

Paralogs and homologs

### **2 Pairwise alignment**

### **3 Global alignment**

### **4 Local alignment**

### **5 Number of possible alignments**

Too many to do by hand – need for automatic tools and rapid alignment methods

## **6 Rapid alignment methods**

### **6.a Introduction**

### **6.b Search space reduction**

### **6.c Binary searches**

### **6.d FASTA**

### **6.e BLAST**

## **7 Multiple alignments**

## **8 Proof of concept**

# 1 The biological problem

## Introduction

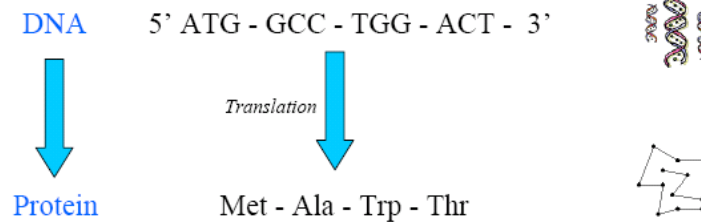
- Much of biology is based on recognition of shared characters among organisms, extending from shared biochemical pathways among eukaryotes to shared skeletal structures among tetrapods.
  - Note;
    - Eukaryotes are organism whose cells contain complex structures enclosed within membranes. Almost all species of large organisms are eukaryotes, including animals, plants and fungi
    - Tetrapods are vertebrate (i.e. with spine) animals having four feet, legs or leglike appendages. Amphibians, reptiles, dinosaurs/birds, and mammals are all tetrapods
- The advent of protein and nucleic acid sequencing in molecular biology made possible comparison of organisms in terms of their DNA or the proteins that DNA encodes.

## Introduction

- These comparisons are important for a number of reasons.
  - First, they can be used to establish evolutionary relationships among organisms using methods analogous to those employed for anatomical characters.
  - Second, comparison may allow identification of functionally conserved sequences (e.g., DNA sequences controlling gene expression).
  - Finally, such comparisons between humans and other species may identify corresponding genes in model organisms, which can be genetically manipulated to develop models for human diseases.

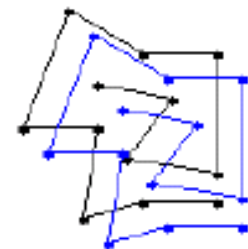
# Introduction

- Hence, in general, there are two bases for sequence alignment
  - Evolutionary
  - Structural



```

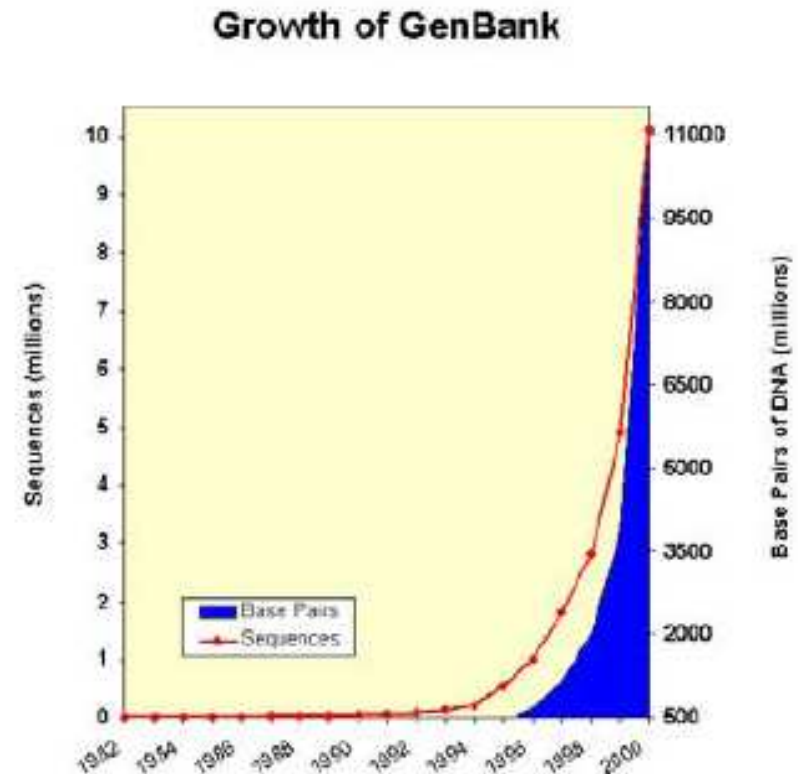
          10      20      30      40      50
M A R Y R C C L T H S G S R C R R R R R R R C R R R R R R F G R R R R R R V C C R R P T V I R C T R Q
: : : : : : : : : : : : : : : : : : : : : : : : : : : : : : : : : : : : : : : : : : : : : : : : : :
S P S I M A R Y R C C R S H S R S R C R P R R R R - C R R R R R R C C P R R R R A V C C R R P T V I R C R R C
          10      20      30      40      50
    
```



(S-star Subbiah)

## Biological sequences and their meaning

- Ever-growing data sequence data bases make available a wealth of data to explore.
  - Recall that these data bases have a tendency of doubling approximately every 14 months and that
  - they comprise a total of over 11 billion bases from more than 100,000 species



(S-star Subbiah)

## Evolutionary basis of alignment

- This lies in the fact that sequence alignment enables the researcher to determine if two sequences display sufficient similarity to justify the inference of homology.
- Understanding the difference between similarity and homology is of utmost importance:
  - Similarity is an observable quantity that may be expressed as a % identity or some other measure.
  - Homology is a conclusion drawn from the data that the two genes share a common evolutionary history.

(S-star Subbiah)



## Evolutionary basis of alignment

- Genes are either homologous or not homologous.
- There are no degrees of homology as are there in similarity.
- While it is presumed that the homologous sequences have diverged from a common ancestral sequence through iterative molecular changes we do not actually know what the ancestral sequence was.
- In contrast to homology, there is another concept called homoplasy. Similar characters that result from independent processes (i.e., convergent evolution) are instances of *homoplasy*

(S-star Subbiah)

## Evolutionary basis of alignment

- Consequently, an alignment just reflects the probable evolutionary history of the two genes for the proteins.
  - Residues that have aligned and are not identical represent substitutions.
  - Regions in which the residues of one sequence correspond to nothing in the other would be interpreted as either an insertion/deletion. These regions are represented in an alignment as gaps.
- Certain regions are more conserved than others. These might point towards crucial residues (structure/function)
- Note that there may be certain regions conserved but not functionally related, due to historical reasons.
  - Especially, from closely related species that have not had sufficient time to diverge.

(S-star Subbiah)

## Homology versus similarity

- Hence, sequence similarity is not sequence homology and can occur by chance ...
- If two sequences have accumulated enough mutations over time, then the similarity between them is likely to be low.
- Consequently, homology is more difficult to detect over greater evolutionary distances

#mutations

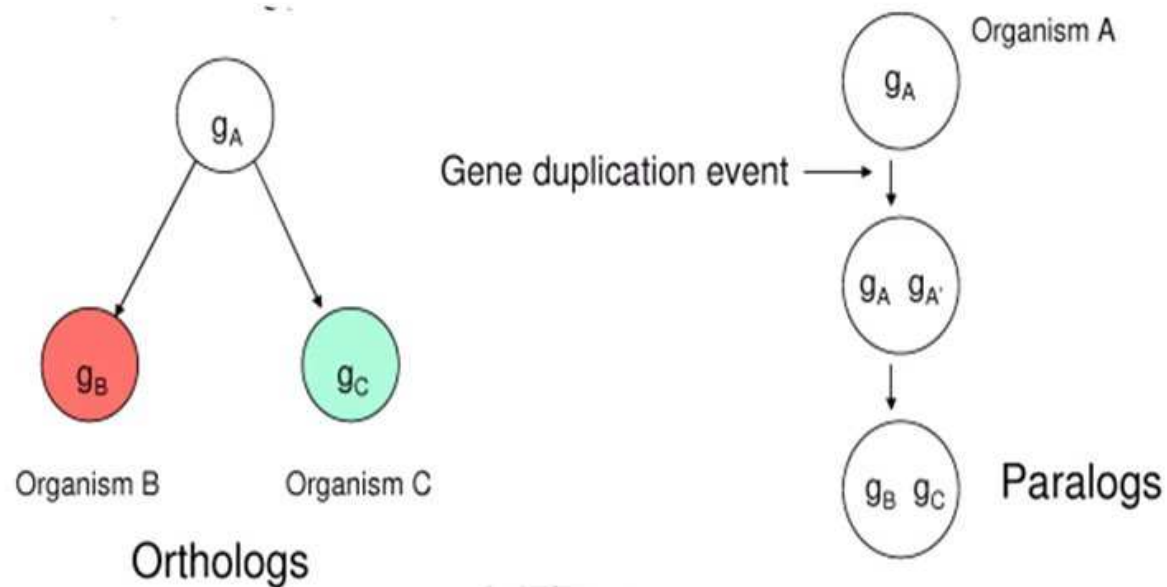
0 agt gt ccgt t aagt gcgt t c  
 1 agt gt ccgt t at agt gcgt t c  
 2 agt gt ccgct t at agt gcgt t c  
 4 agt gt ccgct t aagggcgt t c  
 8 agt gt ccgct t caaggggcgt  
 16 gggccgt t cat gggggt  
 32 gcagggcgt cact gagggct

#mutations

64 acagt ccgt t cgggct at t g  
 128 cagagcact accgc  
 256 cacgagt aagat at agct  
 512 t aat cgt gat a  
 1024 accct t at ct act t cct ggagt t  
 2048 agcgacct gcccaa  
 4096 caaac

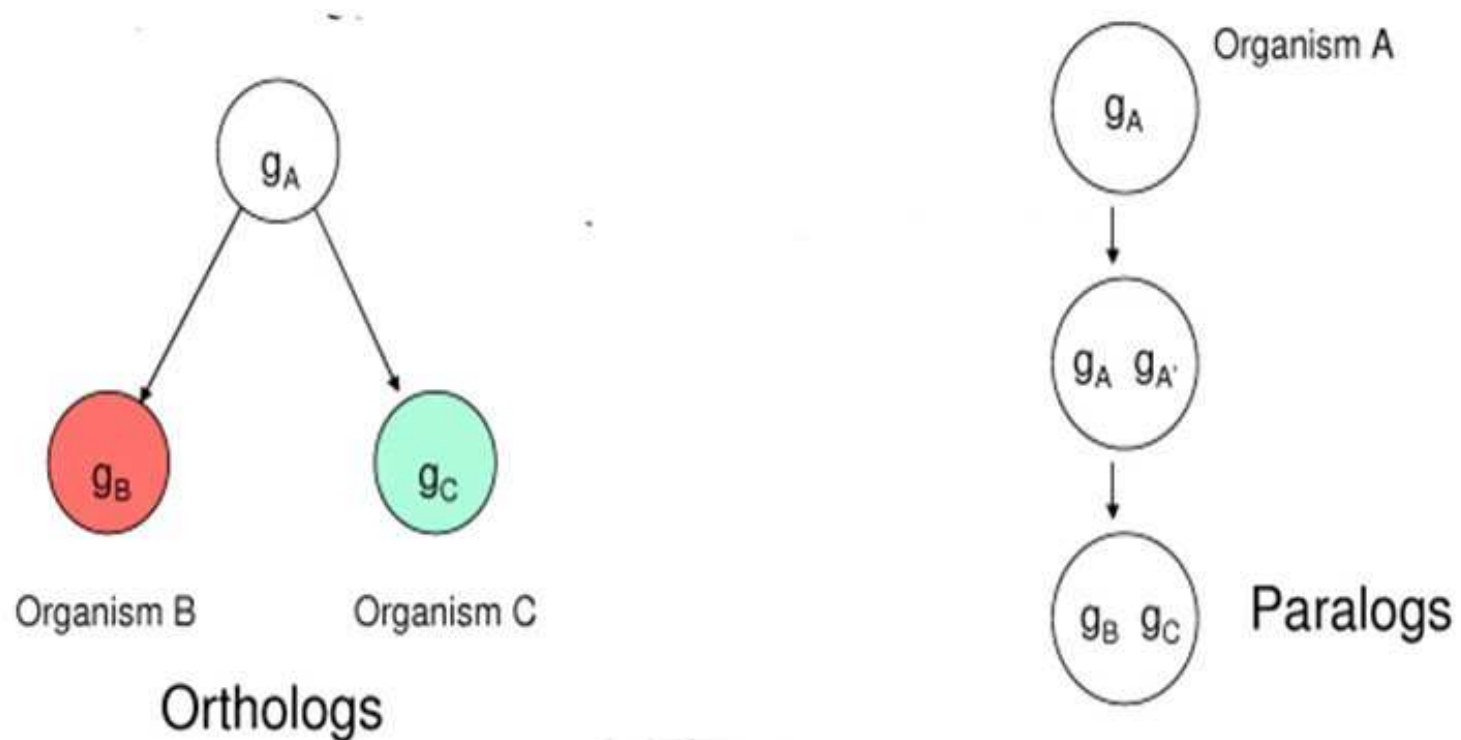
## Orthologs and paralogs

- We distinguish between two types of homology
  - Orthologs: homologs from two different species, separated by a, what is called, a *speciation event*
  - Paralogs: homologs within a species, separated by a *gene duplication event*.



## Orthologs and paralogs

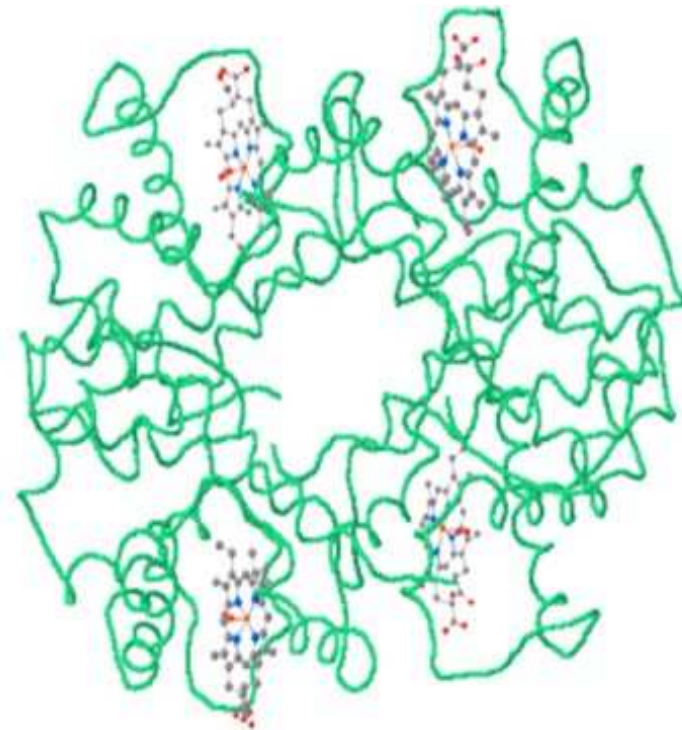
- Orthologs typically retain the original function
- In paralogs, one copy is free to mutate and acquire new function. In other words, there is no selective pressure.



## Example of paralogy: hemoglobin

- Hemoglobin is a protein complex that transports oxygen
  - In humans, hemoglobin consists of 4 protein subunits and 4 non-protein heme groups
  - In adults, 3 types are normally present:
    - Hemoglobin A
    - Hemoglobin A2
    - Hemoglobin F
- each with a different combination of subunits.

- Each subunit is encoded by a separate gene.



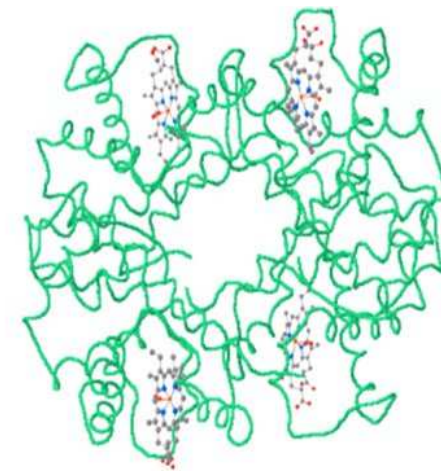
Hemoglobin A,  
[www.rcsb.org/pdb/explore.do?structureId=1GZX](http://www.rcsb.org/pdb/explore.do?structureId=1GZX)

## Example of paralogy: hemoglobin

- The subunit genes are paralogs of each other. In other words, they have a common ancestor gene
- Check out the hemoglobin human paralogs using the NCBI sequence databases:

<http://www.ncbi.nlm.nih.gov/sites/entrez?db=nucleotide>

- Find the 4 human hemoglobin protein subunits (alpha, beta, gamma and delta)
- Compare their sequences

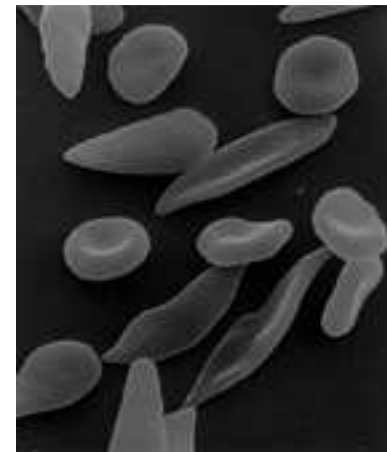


Hemoglobin A,  
[www.rcsb.org/pdb/explore.do?structureId=1GZX](http://www.rcsb.org/pdb/explore.do?structureId=1GZX)

## Sickle cell disease

- Sickle-cell disease, or sickle-cell anaemia (or drepanocytosis), is a life-long blood disorder characterized by red blood cells that assume an abnormal, rigid, sickle shape.
- Sickling decreases the cells' flexibility and results in a risk of various complications.
- The sickling occurs because of a mutation in the hemoglobin gene. Life expectancy is shortened, with studies reporting an average life expectancy of 42 and 48 years for males and females, respectively

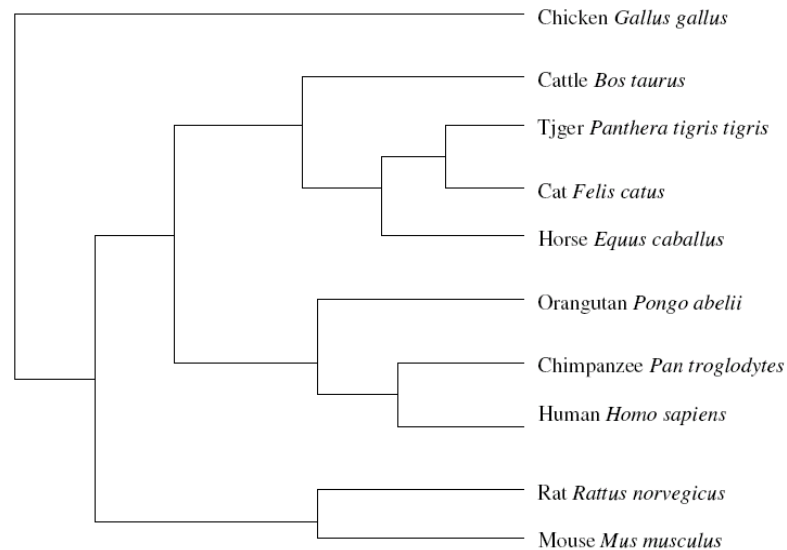
(Wikipedia)





## Example of orthology: insulin

- The genes that code for insulin in humans (*Homo Sapiens*) and mouse (*Mus musculus*) are orthologs
  - They have a common ancestor gene in the ancestor species of human and mouse



(example of a phylogenetic tree - Lakshmi)

## Structural basis for alignment

- It is well-known that when two protein sequences have more than 20-30% identical residues aligned the corresponding 3-D structures are almost always structurally very similar.
- Overall folds are identical and structures differ in detail.
- Form often follows function. So sequence similarity by way of structural similarity implies similar function.
- Therefore, sequence alignment is often an approximate predictor of the underlying 3-D structural alignment

(S-star Subbiah)

## Caveat

- Computational predictions only make suggestions
- To make a conclusive case further experimental tests must validate these suggestions
  - Evolutionary relatedness must be confirmed either by
    - experimental evidence for evolutionary history or
    - experimental establishment of similar function.
  - For structural relatedness the 3-D structures must be experimentally determined and compared.

(S-star Subbiah)

## 2 Pairwise alignment

### Introduction

- An important activity in biology is identifying DNA or protein sequences that are similar to a sequence of experimental interest, with the goal of finding sequence homologs among a list of *similar* sequences.
- By writing the sequence of gene  $g_A$  and of each candidate homolog as strings of characters, with one string above the other, we can determine at which positions the strings do or do not match.
- This is called an alignment. Aligning polypeptide sequences with each other raises a number of additional issues compared to aligning nucleic acid sequences, because of particular constraints on protein structures and the genetic code (not covered in this class).

## Introduction

- There are many different ways that two strings might be aligned. Ordinarily, we expect homologs to have more matches than two randomly chosen sequences.
- The seemingly simple alignment operation is not as simple as it sounds.
- Example (matches are indicated by . and - is placed opposite bases not aligned):

ACGTCTAG	2 matches
..	5 mismatches
ACTCTAG-	1 not aligned

## Introduction

- We might instead have written the sequences

ACGTCTAG	5 matches
.....	2 mismatches
-ACTCTAG	1 not aligned

- We might also have written

ACGTCTAG	7 matches
.. .....	0 mismatches
AC-TCTAG	1 not aligned

Which alignment is better?  
What does better mean?

## Introduction

- Next, consider aligning the sequence TCTAG with a long DNA sequence:

```
...AACTGAGTTTACGCTCATAGA...  
      T---CT-A--G
```

- We might suspect that if we compared any string of modest length with another very long string, we could obtain perfect agreement if we were allowed the option of "not aligning" with a sufficient number of letters in the long string.
- Clearly, we would prefer some type of parsimonious alignment. One that does not postulate an excessive number of letters in one string that are not aligned opposite identical letters in the other.

## Introduction

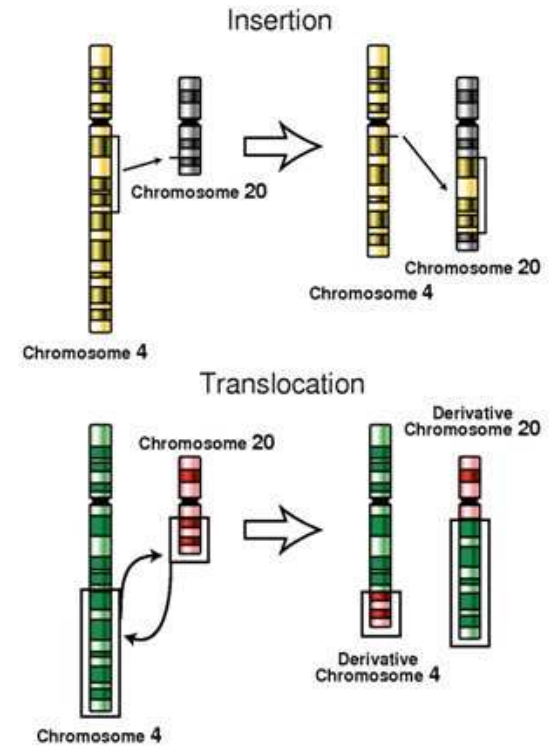
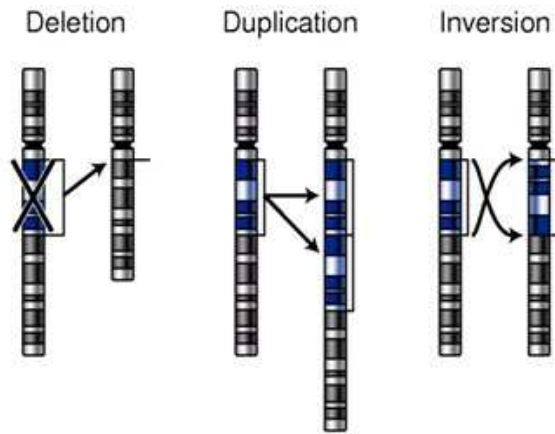
- We have seen that there are multiple ways of aligning two sequence strings, and we may wish to compare our target string (target meaning the given sequence of interest) to entries in databases containing more than  $10^7$  sequence entries or to collections of sequences billions of letters long.
- How do we do this?
  - We differentiate between alignment of two sequences with each other
    - which can be done using the entire strings (global alignment )
    - or by looking for shorter regions of similarity contained within the strings that otherwise do not significantly match (local alignment)
  - and multiple-sequence alignment (alignment of more than two strings)



## Introduction

- The approach adopted is guided by biology
  - It is possible for evolutionarily related proteins and nucleic acids to display substitutions at particular positions (resulting from known mutational processes).
  - Also, it is possible for these to display insertions or deletions (less likely than substitution).
  - In total, the DNA sequence can be modified by several biological processes (cfr next slide)

# Types of mutations



NIH – National Human Genome Research Institute)

## Introduction

- Mutations such as segmental duplication, inversion, translocation often involve DNA segments larger than the coding regions of genes.
- They usually do not affect the type of alignment that we are currently interested in
- Point mutations, insertion or deletion of short segments are important in aligning targets whose size are less than or equal to the size of coding regions of genes, and need to be explicitly acknowledged in the alignment process.

## Introduction

- Consider again the following alignment

ACGTCTAG	7 matches
.. . . . .	0 mismatches
AC-TCTAG	1 not aligned

- We can't tell whether the string at the top resulted from the insertion of G in ancestral sequence ACTCTAG or whether the sequence at the bottom resulted from the deletion of G from ancestral sequence ACGTCTAG.
- For this reason, alignment of a letter opposite nothing is simply described as an indel.

## Toy example

- Suppose that we wish to align WHAT with WHY.
- Our *goal* is to find the highest-scoring alignment. This means that we will have to devise a scoring system to characterize each possible alignment.
- One possible alignment solution is

WHAT  
WH-Y

- However, we need a rule to tell us how to calculate an alignment score that will, in turn, allow us to identify which alignment is best.
- Let's use the following scores for each instance of match, mismatch, or indel:
  - identity (match)                    +1
  - substitution (mismatch)           - $\mu$
  - indel                                        -  $\delta$

## Toy example

- The minus signs for substitutions and indels assure that alignments with many substitutions or indels will have low scores.
- We can then define the score  $S$  as the sum of individual scores at each position:

$$S(\text{WHAT}/\text{WH} - \text{Y}) = 1 + 1 - \delta - \mu$$

- There is a more general way of describing the scoring process (not necessary for "toy" problems such as the one above).
- In particular: write the target sequence (WHY) and the search space (WHAT) as rows and columns of a matrix:

## Toy example

	-	W	H	A	T
-					
W		x			
H			x	x	
Y					x

- We have placed an x in the matrix elements corresponding to a particular alignment
- We have included one additional row and one additional column for initial indels (-) to allow for the possibility (not applicable here) that alignments do not start at the initial letters (W opposite W in this case).

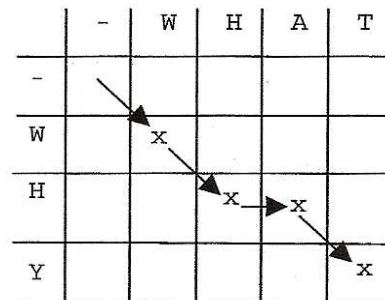
## Toy example

- We can indicate the alignment

WHAT  
WH-Y

as a path through elements of the matrix (arrows).

	-	W	H	A	T
-					
W		x			
H			x	x	
Y					x



- If the sequences being compared were identical, then this path would be along the diagonal.



## Toy example

- Other alignments of WHAT with WHY would correspond to paths through the matrix other than the one shown.
- Each step from one matrix element to another corresponds to the incremental shift in position along one or both strings being aligned with each other, and we could write down in each matrix element the running score up to that point instead of inserting x.

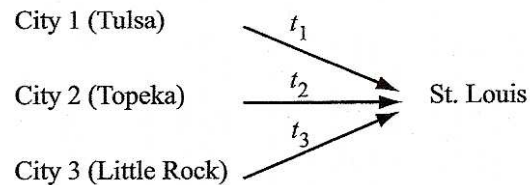
	-	W	H	A	T
-	0				
W		1			
H			2	$2-\delta$	
Y					$2-\delta-\mu$

## Toy example

- What we seek is the path through the matrix that produces the greatest possible score in the element at the lower right-hand corner.
- That is our "destination," and it corresponds to having used up all of the letters in the search string (first column) and search space (first row)-this is the meaning of *global alignment*.
- Using a scoring matrix such as this employs a particular trick of thinking.

## Toy example

- For example, what is the "best" driving route from Los Angeles to St. Louis? We could plan our trip starting in Los Angeles and then proceed city to city considering different routes. For example, we



might go through Phoenix, Albuquerque, Amarillo, etc., or we could take a more northerly route through Denver. We seek an itinerary (best route) that minimizes the driving time. One way of analyzing alternative routes is to consider the driving time to a city relatively close to St. Louis and add to it the driving time from that city to St. Louis.

## Toy example

- We would recognize that the best route to St. Louis is the route to St. Louis from city  $n$  + the best route to  $n$  from Los Angeles.
- If  $D_1$ ,  $D_2$ , and  $D_3$  are the driving times to cities 1, 2, and 3 from Los Angeles, then the driving times to St. Louis through these cities are given by  $D_1 + t_1$ ,  $D_2 + t_2$ , and  $D_3 + t_3$ .
- Suppose that the driving time to St. Louis through Topeka (City 2) turned out to be smaller than the times to St. Louis through Tulsa or Little Rock (i.e.,  $D_2 + t_2$  were the minimum of  $\{D_1 + t_1, D_2 + t_2, D_3 + t_3\}$ ). We then know that we should travel through Topeka.
- We next ask how we get to Topeka from three prior cities, seeking the route that minimizes the driving time to City 2.

## Toy example

- Analyzing the best alignment using an alignment matrix proceeds similarly, first filling in the matrix by working forward and then working backward from the "destination" (last letters in a global alignment) to the starting point.
- This general approach to problem-solving is called *dynamic programming*.

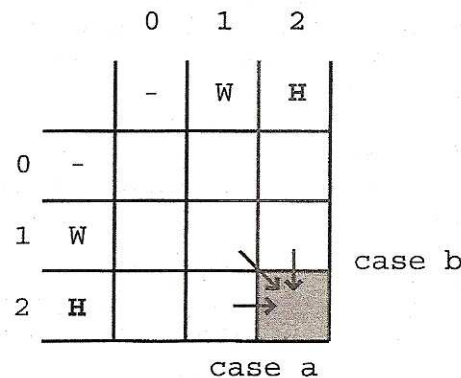
## Dynamic programming

- Dynamic programming; a computer algorithmic technique invented in the 1940's.
- Dynamic programming (DP) has applications to many types of problems.
- Key properties of problems solvable with DP include that the optimal solution typically contains optimal solutions to subproblems, and only a “small” number of subproblems are needed for the optimal solution.

(T.H. Cormen et al., Introduction to Algorithms, McGraw-Hill 1990).

## Toy example (continued)

- The best alignment is revealed by beginning at the destination (lower right-hand corner matrix element) and working backward, identifying the path that maximizes the score at the end.



- To do this, we will have to calculate scores for all possible paths. into each matrix element ("city") from its neighboring elements above, to the left, and diagonally above.

## Toy example

- To illustrate, suppose that we want to continue an ongoing alignment process using WHAT and WHY and that we have gotten to the point at which we want to continue the alignment into the shaded element of the matrix below.
- We have now added row and column numbers to help us keep track of matrix elements.

	0	1	2	
	-	W	H	
0	-			
1	W			case b
2	H			

case a



## Toy example

- There are three possible paths into element (3, 3) (aligning left to right with respect to both strings; letters not yet aligned are written in parentheses):

Case a.

If we had aligned WH in WHY with W in WHAT (corresponding to element (2, 1)), adding H in WHAT without aligning it to H in WHY corresponds to an insertion of H (relative to WHY) and advances the alignment from element (2, 1) to element (2,2) (horizontal arrow):

(W) H (AT)  
(WH) - (Y)

## Toy example

Case b.

If we had aligned W in WHY with WH in WHAT (corresponding to element (1, 2)), adding the H in WHY without aligning it to H in WHAT corresponds to insertion of H (relative to WHAT) and advances the alignment from element (1, 2) to element (2, 2) (vertical arrow):

(WH)-(AT)  
(W)H (Y)

## Toy example

Case c.

If we had aligned W in WHY with W in WHAT (corresponding to element (1,1)), then we could advance to the next letter in both strings, advancing the alignment from (1,1) to (2,2) (diagonal arrow above):

(W)H(AT)  
(W)H (y)

- Note that horizontal arrows correspond to adding indels to the string written vertically and that vertical arrows correspond to adding indels to the string written horizontally.

## Toy example

- Associated with each matrix element  $(x, y)$  from which we could have come into  $(2,2)$  is the score  $S_{x,y}$  up to that point.
- Suppose that we assigned scores based on the following scoring rules:

identity (match)	+1
substitution (mismatch)	-1
indel	-2

- Then the scores for the three different routes into  $(2,2)$  are

$$\text{Case a : } S_{2,2} = S_{2,1} - 2,$$

$$\text{Case b : } S_{2,2} = S_{1,2} - 2,$$

$$\text{Case c : } S_{2,2} = S_{1,1} + 1.$$

## Toy example

- The path of the cases a, b, or c that yields the highest score for  $S_{2,2}$  is the preferred one, telling us which of the alignment steps is best.
- Using this procedure, we will now go back to our original alignment matrix and fill in all of the scores for all of the elements, *keeping track of the path into each element that yielded the maximum score to that element.*
- The initial row and column labeled by (-) corresponds to sliding WHAT or WHY incrementally to the left of the other string without aligning against any letter of the other string.
- Aligning (-) opposite (-) contributes nothing to the alignment of the strings, so element (0,0) is assigned a score of zero.
- Since penalties for indels are -2, the successive elements to the right or down from element (0, 0) each are incremented by -2 compared with the previous one.

## Toy example

- Thus  $S_{0,1}$  is -2, corresponding to W opposite -,  $S_{0,2}$  is -4, corresponding to WH opposite -, etc., where the letters are coming from WHAT.
- Similarly,  $S_{1,0}$  is -2, corresponding to - opposite W,  $S_{2,0}$  is -4, corresponding to - opposite WH, etc., where the letters are coming from WHY.
- The result up to this point is

		0	1	2	3	4
	-	0	-2	-4	-6	-8
0	-	0	-2	-4	-6	-8
1	W	-2				
2	H	-4				
3	Y	-6				

## Toy example

- Now we will calculate the score for (1,1). This is the greatest of  $S_{0,0} + 1$  (W matching W, starting from (0, 0)),  $S_{0,1} - 2$ , or  $S_{1,0} - 2$ .
- Clearly,  $S_{0,0} + 1 = 0 + 1 = 1$  "wins". (The other sums are  $-2 - 2 = -4$ .)
- We record the score value +1 in element (1, 1) and record an arrow that indicates where the score came from.

		0	1	2	3	4
	-		W	H	A	T
0	-	0	-2	-4	-6	-8
1	W	-2	+1			
2	H	-4	-1			
3	Y	-6				

## Toy example

- The same procedure is used to calculate the score for element (2,1) (see above).
- Going from (1,0) to (2,1) implies that H from WHY is to be aligned with W of WHAT (after first having aligned W from WHY with (-)). This would correspond to a substitution, which contributes -1 to the score. So one possible value of  $S_{2,1} = S_{1,0} - 1 = -3$ .
- But (2, 1) could also be reached from (1, 1), which corresponds to aligning H in WHY opposite an indel in WHAT (i.e., not advancing a letter in WHAT). From that direction,  $S_{2,1} = S_{1,1} - 2 = 1 - 2 = -1$ .
- Finally, (2, 1) could be entered from (2,0), corresponding to aligning W in WHAT with an indel coming after H in WHY. In that direction,  $S_{2,1} = S_{2,0} - 2 = -4 - 2 = -6$ .
- We record the maximum score into this cell ( $S_{2,1} = S_{1,1} - 2 = -1$ ) and the direction from which it came.



## Toy example

- The remaining elements of the matrix are filled in by the same procedure, with the following result:

	0	1	2	3	4	
	-	W	H	A	T	
0	-	0	-2	-4	-6	-8
1	W	-2	+1	-1	-3	-5
2	H	-4	-1	2	0	-2
3	Y	-6	-3	0	1	-1

Arrows in the matrix indicate the path from (0,0) to (3,4): (0,0) to (1,1), (1,1) to (2,2), (2,2) to (3,3), and (3,3) to (3,4). Additionally, arrows point from (1,2) to (2,2) and from (2,3) to (3,3), indicating alternative paths to the cell (3,4).

- The final score for the alignment is  $S_{3,4} = -1$ .
- The score could have been achieved by either of two paths (implied by two arrows into (3, 4) yielding the same score).

## Toy example

- The path through element (2,3) (upper path, bold arrows) corresponds to the alignment

WHAT  
WH-Y

which is read by tracing back through all of the elements visited in that path.

- The lower path (through element (3, 3)) corresponds to the alignment

WHAT  
WHY-

- Each of these alignments is equally good (two matches, one mismatch, one indel).

## Toy example

- Note that we always recorded the score for the best path into each element.
- There are paths through the matrix corresponding to very "bad" alignments. For example, the alignment corresponding to moving left to right along the first row and then down the last column is

```
WHAT - -  
----WHY
```

with score -14.

- For this simple problem, the computations were not tough. But when the problems get bigger, there are so many different possible alignments that an organized approach is essential.
- Biologically interesting alignment problems are far beyond what we can handle with a No. 2 pencil and a sheet of paper, like we just did.

## 3 Global alignment

### Formal development

- We are given two strings, not necessarily of the same length, but from the same alphabet:

$$A = a_1 a_2 a_3 \cdots a_n,$$
$$B = b_1 b_2 b_3 \cdots b_m.$$

- Alignment of these strings corresponds to consecutively selecting each letter or inserting an indel in the first string and matching that particular letter or indel with a letter in the other string, or introducing an indel in the second string to place opposite a letter in the first string.

## Formal development

- Graphically, the process is represented by using a matrix as shown below for  $n = 3$  and  $m = 4$ :

	0	1	2	3	4
	-	$b_1$	$b_2$	$b_3$	$b_4$
0	-	→			
1	$a_1$		↘	→	
2	$a_2$				↘
3	$a_3$				↓

- The alignment corresponding to the path indicated by the arrows is

$$\begin{array}{cccccc}
 b_1 & b_2 & b_3 & b_4 & - & \\
 - & a_1 & - & a_2 & a_3 & 
 \end{array}$$

## Formal development

- Any alignment that can be written corresponds to a unique path through the matrix.
- The quality of an alignment between  $A$  and  $B$  is measured by a score,  $S(A, B)$ , which is large when  $A$  and  $B$  have a high degree of similarity.
  - If letters  $a_i$  and  $b_j$  are aligned opposite each other and are the same, they are an instance of an **identity**.
  - If they are different, they are said to be a **mismatch**.
- The score for aligning the first  $i$  letters of  $A$  with the first  $j$  letters of  $B$  is

$$S_{i,j} = S(a_1a_2 \cdots a_i, b_1b_2 \cdots b_j).$$

## Formal development

- $S_{i,j}$  is computed recursively as follows. There are three different ways that the alignment of  $a_1 a_2 \dots a_i$  with  $b_1 b_2 \dots b_j$  can end:

$$\begin{array}{ll} \text{Case a:} & (a_1 \ a_2 \ \dots \ a_i) \quad - \\ & (b_1 \ b_2 \ \dots \ b_{j-1}) \ b_j, \\ \text{Case b:} & (a_1 \ a_2 \ \dots \ a_{i-1}) \ a_i \\ & (b_1 \ b_2 \ \dots \ b_j) \quad -, \\ \text{Case c:} & (a_1 \ a_2 \ \dots \ a_{i-1}) \ a_i \\ & (b_1 \ b_2 \ \dots \ b_{j-1}) \ b_j, \end{array}$$

where the inserted spaces "-" correspond to insertions or deletions ("indels") in  $A$  or  $B$ .

- Scores for each case are defined as follows:

$$\begin{aligned} s(a_i, b_j) &= \text{score of aligning } a_i \text{ with } b_j \\ & \quad (= +1 \text{ if } a_i = b_j, -\mu \leq 0 \text{ if } a_i \neq b_j, \text{ for example}), \\ s(a_i, -) &= s(-, b_j) = -\delta \leq 0 \text{ (for indels)}. \end{aligned}$$

## Formal development

- With global alignment, indels will be added as needed to one or both sequences such that the resulting sequences (with indels) have the same length. The best alignment up to positions  $i$  and  $j$  corresponds to the case a, b, or c before that produces the largest score for  $S_{i,j}$ :

$$S_{i,j} = \max \left\{ \begin{array}{l} S_{i-1,j-1} + s(a_i, b_i) \\ S_{i-1,j} - \delta \\ S_{i,j-1} - \delta \end{array} \right\} \quad \begin{array}{l} \text{Case c} \\ \text{Case b.} \\ \text{Case a} \end{array}$$

- The "max" indicates that the one of the three expressions that yields the maximum value will be employed to calculate  $S_{i,j}$



## Formal development

- Except for the first row and first column in the alignment matrix, the score at each matrix element is to be determined with the aid of the scores in the elements immediately above, immediately to the left, or diagonally above and to the left of that element.
- The scores for elements in the first row and column of the alignment matrix are given by

$$S_{i,0} = -i\delta, \quad S_{0,j} = -j\delta.$$

- The score for the best global alignment of  $A$  with  $B$  is  $S(A, B) = S_{n,m}$  and it corresponds to the highest-scoring path through the matrix and ending at element  $(n, m)$ . It is determined by tracing back element by element along the path that yielded the maximum score into each matrix element.

## Exercise

- What is the maximum score and corresponding alignment for aligning

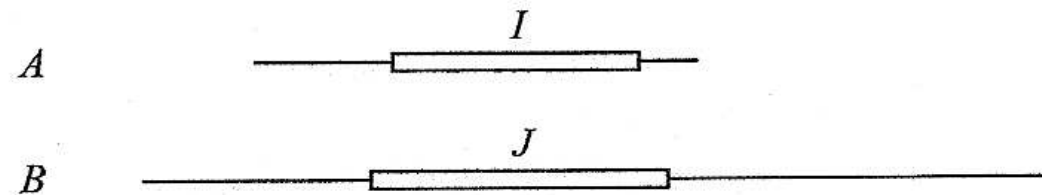
A=ATCGT with B=TGGTG?

- For scoring, take
  - $s(a_i, b_j) = +1$  if  $a_i = b_j$ ,
  - $s(a_i, b_j) = -1$  if  $a_i \neq b_j$  and
  - $s(a_i, -) = s(-, b_j) = -2$

## 4 Local alignment

### Rationale

- Proteins may be multifunctional. Pairs of proteins that share one of these functions may have regions of similarity embedded in otherwise dissimilar sequences.
- For example, human TGF- $\beta$  receptor (which we will label *A*) is a 503 amino acid (aa) residue protein containing a protein kinase domain extending from residue 205 through residue 495. This 291 aa residue segment of TGF- $\beta$ receptor is similar to an interior 300 aa residue portion of human bone morphogenic protein receptor type II precursor

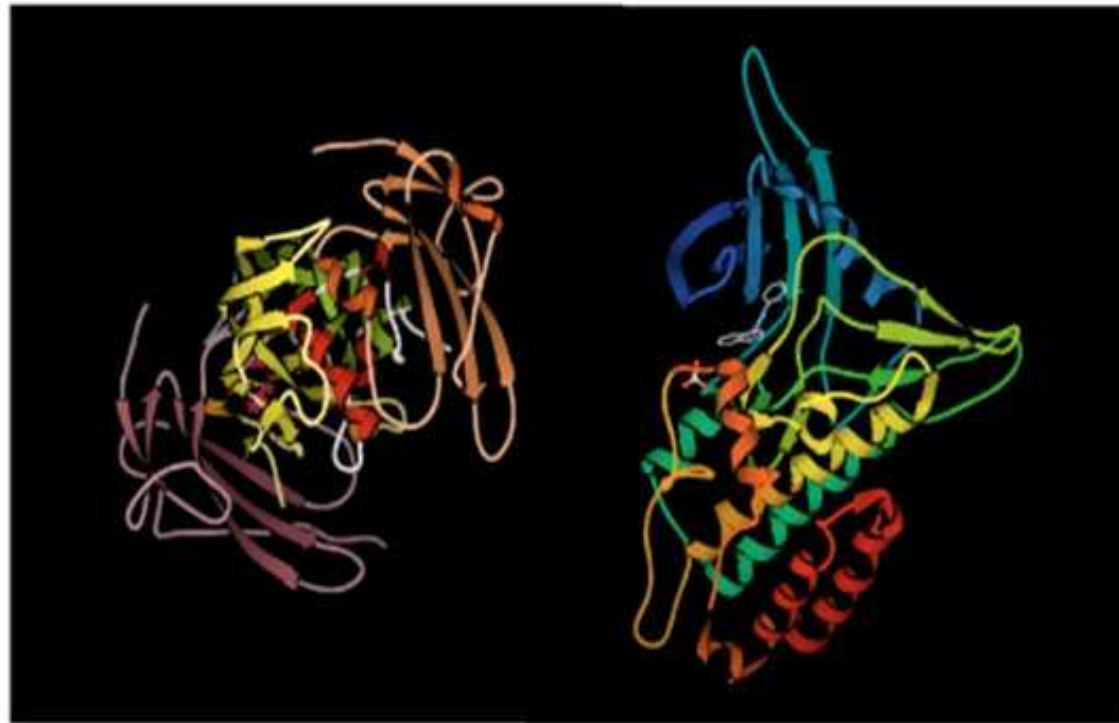


(which we will label *B*), a polypeptide that is 1038 aa residues long.

## Rationale

Human bone morphogenic protein receptor type II precursor (left) has a 300 aa region that resembles 291 aa region in TGF- $\beta$  receptor (right).

The shared function here is protein kinase.



## Rationale

- With only partial sequence similarity and very different lengths, attempts at global alignment of two sequences such as these would lead to huge cumulative indel penalties.
- What we need is a method to produce the best *local alignment*; that is, an alignment of segments contained *within* two strings (Smith and Waterman, 1981).
- As before, we will need an alignment matrix, and we will seek a high-scoring path through the matrix.
- Unlike before, the path will traverse only *part* of the matrix. Also, we do not apply indel penalties if strings *A* and *B* fail to align at the ends.

## Rationale

- Hence, instead of having elements  $-i\delta$  and  $-j\delta$  in the first row and first column, respectively ( $-\delta$  being the penalty for each indel), all the elements in the first row and first column will now be zero.
- Moreover, since we are interested in paths that yield high scores over stretches less than or equal to the length of the smallest string, there is no need to continue paths whose scores become too small.
  - If the best path to an element from its immediate neighbors above and to the left (including the diagonal) leads to a negative score, we will arbitrarily assign a 0 score to that element.
  - We will identify the best local alignment by tracing back from the matrix element having the highest score. This is usually not (but occasionally may be) the element in the lower right-hand corner of the matrix.

## Mathematical formulation

- We are given two strings  $A = a_1a_2a_3 \dots a_n$  and  $B = b_1b_2b_3 \dots b_m$
- Within each string there are intervals  $I$  and  $J$  that have similar sequences.  $I$  and  $J$  are *intervals* of  $A$  and  $B$ , respectively. We indicate this by writing  $I \subset A$  and  $J \subset B$ , where “ $\subset$ ” means “is an interval of.”
- The best local alignment score,  $M(A, B)$ , for strings  $A$  and  $B$  is

$$M(A, B) = \max\{S(I, J) : I \subset A, J \subset B\}$$

where  $S(I, J)$  is the score for subsequences  $I$  and  $J$  and  $S(\emptyset, \emptyset) = 0$ .

- Elements of the alignment matrix are  $M_{i,j}$ , and since we are not applying indel penalties at the ends of  $A$  and  $B$ , we write

$$M_{i,0} = M_{0,i} = 0.$$

## Mathematical formulation

- The score up to and including the matrix element  $M_{i,j}$  is calculated by using scores for the elements immediately above and to the left (including the diagonal), but this time scores that fall below zero will be replaced by zero.
- The scoring for matches, mismatches, and indels is otherwise the same as for global alignment.
- The resulting expression for scoring  $M_{i,j}$  is

$$M_{i,j} = \max \left\{ \begin{array}{l} M_{i-1,j-1} + s(a_i, b_i) \\ M_{i-1,j} - \delta \\ M_{i,j-1} - \delta \\ 0 \end{array} \right\}.$$



## Mathematical formulation

- The best local alignment is the one that ends in the matrix element having the highest score:

$$\max\{S(I, J) : I \subset A, J \subset B\} = \max_{i,j} M_{i,j}.$$

- Thus, the best local alignment score for strings  $A$  and  $B$  is

$$M(A, B) = \max_{i,j} M_{i,j}.$$

## Exercise

- Determine the best local alignment and the maximum alignment score for

A=ACCTAAGG and B=GGCTCAATCA

- For scoring, take
  - $s(a_i, b_j) = +2$  if  $a_i = b_j$ ,
  - $s(a_i, b_j) = -1$  if  $a_i \neq b_j$  and
  - $s(a_i, -) = s(-, b_j) = -2$

## Scoring rules

- We have used alignments of nucleic acids as illustrative examples, but it should be noted that for protein alignments the scoring is much more complicated.
- Hence, at this point, we address briefly the issue of assigning appropriate values to  $s(a_i, b_j)$ ,  $s(a_i, -)$ , and  $s(-, b_j)$  for nucleotides.
- For scoring issues in case of amino acids, please refer to Deonier et al. 2005 (Chapter 7, p182-189).

## Scoring rules

- Considering  $s(a_i, b_j)$  first, we write down a **scoring matrix** containing *all* possible ways of matching  $a_i$  with  $b_j$ ,  $a_i, b_j \in \{A, C, G, T\}$  and write in each element the scores that we have used for matches +1 and

		$b_j:$			
		A	C	G	T
$a_i:$	A	1	-1	-1	-1
	C	-1	1	-1	-1
	G	-1	-1	1	-1
	T	-1	-1	-1	1

mismatches -1.

- Note that this scoring matrix contains the assumption that aligning A with G is just as bad as aligning A with T because the mismatch penalties are the same in both cases.

## Scoring rules

- A first issue arises when observing that studies of mutations in homologous genes have indicated that transition mutations ( $A \rightarrow G$ ,  $G \rightarrow A$ ,  $C \rightarrow T$ , or  $T \rightarrow C$ ) occur approximately twice as frequently as do transversions ( $A \rightarrow T$ ,  $T \rightarrow A$ ,  $A \rightarrow C$ ,  $G \rightarrow T$ , etc.). (A, G are purines, larger molecules than pyrimidines T, C)
- Therefore, it may make sense to apply a lesser penalty for transitions than for transversions
- The collection of  $s(a_i, b_j)$  values in that case might be represented as

		$b_j :$			
		A	C	G	T
$a_i :$	A	1	-1	-0.5	-1
	C	-1	1	-1	-0.5
	G	-0.5	-1	1	-1
	T	-1	-0.5	-1	1

## Scoring rules

- A second issue relates to the scoring of gaps (a succession of indels), in that sense that we never bothered about whether or not indels are actually independent.
- Up to now, we have scored a gap of length  $k$  as

$$\omega(k) = -k\delta$$

- However, insertions and deletions sometimes appear in "chunks" as a result of biochemical processes such as replication slippage at microsatellite repeats.
- Also, deletions of one or two nucleotides in protein-coding regions would produce frameshift mutations (usually non-functional), but natural selection might allow small deletions that are integral multiples of 3, which would preserve the reading frame and some degree of function.

## Scoring rules

- The aforementioned examples suggest that it would be better to have gap penalties that are not simply multiples of the number of indels.
- One approach is to use an expression such as

$$\omega(k) = -\alpha - \beta(k - 1)$$

- This would allow us to impose a larger penalty for opening a gap ( $-\alpha$ ) and a smaller penalty for gap extension ( $-\beta$  for each additional base in the gap).

## 5 Number of possible global alignments

### Introduction

- For strings of lengths  $n$  and  $m$ , we had to compute three scores going into “inner cells” of a  $(n+1)(m+1)$  matrix and to take a maximum. This implies a computation time of  $O(mn)$ .
  - The computational time complexity of an algorithm with input data size  $n$  is measured in “big O” notation and written as  $O(g(n))$  if the algorithm can be executed in time (or number of steps) less than or equal to  $Cg(n)$  for some constant  $C$ .
- How many possible global alignments are there for two strings of lengths  $m$  and  $n$ ?
  - This is the same thing as asking how many different paths there are through the alignment matrix (excluding backward moves along the strings).



## Number of possible global alignments

- The number of matched pairs will be less than or equal to the smaller of  $m$  and  $n$ .
- We can count the number of alignments,  $\#A$ , by summing the number of alignments having one matched pair, the number of alignments having two matched pairs, and so on up to  $\min(m, n)$  matched pairs.
- Examples of some of the 12 alignments of  $A = a_1a_2a_3a_4$  and  $B = b_1b_2b_3$  having one matched pair are

$$\begin{array}{cccccc}
 - & - & a_1 & a_2 & a_3 & a_4 \\
 b_1 & b_2 & b_3 & - & - & -
 \end{array}
 \quad
 \begin{array}{cccccc}
 - & a_1 & a_2 & a_3 & a_4 & - \\
 b_1 & b_2 & - & - & - & b_3
 \end{array}
 \quad
 \begin{array}{cccccc}
 - & a_1 & - & a_2 & a_3 & a_4 \\
 b_1 & - & b_2 & - & - & b_3
 \end{array}$$

## Number of possible global alignments

- To count the number of ways of having  $k$  aligned pairs, we must choose  $k$  letters from each sequence. From  $A$  this can be done in  $\binom{n}{k}$  ways, and from  $B$  this can be done in  $\binom{m}{k}$  ways.
- Therefore

$$\begin{aligned}
 \#A &= \sum_{k=0}^{\min\{m,n\}} (\# \text{ of alignments having exactly } k \text{ matched pairs}) \\
 &= \sum_{k=0}^{\min\{m,n\}} \binom{n}{k} \binom{m}{k} \\
 &= 1 + nm + \frac{n!}{(n-2)!2!} \times \frac{m!}{(m-2)!2!} + \dots
 \end{aligned}$$

Where does the “1” come in the previous expression?

Where does the “mn” come from?

## Number of possible global alignments

- The result turns out to have a simple expression:

$$\#A = \sum_{k=0}^{\min\{m,n\}} \binom{n}{k} \binom{m}{k} = \binom{n+m}{\min(n,m)}.$$

Can you prove this equality?

- The latter equality requires some manipulation (cfr Deonier et al 2005, p 159-160). The number of global alignments in the special case  $m = n$ ,

$$\#A = \binom{2n}{n} = (2n)!/(n!)^2.$$

can be approximated by using Stirling's approximation.

## Number of possible global alignments

- When we apply Stirling's approximation

$$x! \sim (2\pi)^{1/2} x^{(x+1/2)} e^{-x}.$$

we obtain

$$\#A \sim 2^{2n} / \sqrt{\pi n}.$$

- This approximate value for the number of alignments can also be rationalized in the following simple manner.
  - We are given two strings of equal length,  $A = a_1 a_2 \dots a_n$  and  $B = b_1 b_2 \dots b_n$ .
  - For each of the letters in  $A$ , we have two choices: align it opposite a letter in  $B$  or add an indel. This makes  $2^n$  ways of handling the letters in  $A$ . Similarly for  $B$ . Since the decision "align or add indel" is made for every letter in both strings, the total number of choices for both strings is  $2^n \times 2^n = 2^{2n}$ .

## Number of possible global alignments

- For longer strings, the number of alignments gets very large very rapidly.
  - For  $n = m = 10$ , the number of alignments is already 184,756.
  - For  $n = m = 20$ , the number of alignments is  $1.38 \times 10^{11}$ .
  - For  $m = n = 100$ , there are  $\sim 2^{200}$  possible alignments.
  - In more familiar terms (using  $\log_{10}x = \log_2 x \log_{10}2$ ),  $\log_{10}(2^{200}) = \log_2(2^{200}) \times \log_{10}(2) = 200 \times (0.301) \approx 60$ .

## Number of possible global alignments

- In other words,  $\#A$  when aligning two strings of length 100 is about  $10^{60}$ . This is an astronomically large number.
  - For example, the sun weighs  $1.99 \times 10^{33}$  grams. Each gram contains roughly  $12 \times 10^{23}$  protons and electrons, which means that the sun contains about  $24 \times 10^{56}$  elementary particles.
  - It would take 400 stars the size of our sun to contain as many elementary particles as there are alignments between two equal-length strings containing 100 characters.
- Imagine the **number of local alignments** ....?
- Clearly, we need to have ways of further simplifying the alignment process beyond the  $O(nm)$  method used before ...

## 6 Rapid alignment methods

### 6.a Introduction

- The need for automatic (and rapid!) approaches also arises from the interest in comparing multiple sequences at once (and not just 2)
- Consider a set of n sequences:
  - Orthologous sequences from different organisms
  - Paralogs from multiple duplications

How can we study relationships between these sequences?

```
aggcgagct gcgagt gct a
cgt t agat t gacgct gac
t t ccggct gcgac
gacacggcgaacgga
agt gt gcccgacgagcgaggac
gcgggct gt gagcgct a
aagcggcct gt gt gccct a
at gct gct gccagt gt a
agt cgagccccgagt gc
agt ccgagt cc
act cggt gc
```

## Introduction

- Recall that  $A = a_1a_2 \dots a_i$  and  $B = b_1b_2 \dots b_j$  have alignments that can end in three possibilities:  $(-,b_j)$ ,  $(a_i,b_j)$ , or  $(a_i, -)$ .
- The number of alternatives for aligning  $a_i$  with  $b_j$  can be calculated as  $3 = 2^2 - 1$
- If we now introduce a third sequence  $c_1c_2\dots c_k$ , the three-sequence alignment can end in one of seven ways ( $7 = 2^3 - 1$ ):  $(a_i, -, -)$ ,  $(-, b_j, -)$ ,  $(-, -, c_k)$ ,  $(-, b_j, c_k)$ ,  $(a_i, -, c_k)$ ,  $(a_i, b_j, -)$ , and  $(a_i, b_j, c_k)$ . In other words, the alignment ends in 0, 1, or 2 indels.
- This fundamental term in the recursion is no problem, except that it must be done in time and space proportional to the number of  $(i,j,k)$  positions; that is the product of the length of the sequences  $i \times j \times k$ .
- Hence, solving the recursion using 3-dimensional dynamic programming matrices involves  $O(ijk)$  time and space.



## 6.b Search space reduction

- We can reduce the search space by analyzing word content (see Chapter 4). Suppose that we have the query string I indicated below:

I: TGATGATGAAGACATCAG

- This can be broken down into its constituent set of overlapping k-tuples. For k = 8, this set is

```
TGATGATG
GATGATGA
ATGATGAA
TGATGAAG
.
.
.
GACATCAG
```

## Search space reduction

- If a string is of length  $n$ , then there are  $n - k + 1$   $k$ -tuples that are produced from the string. If we are comparing string  $I$  to another string  $J$  (similarly broken down into words), the absence of anyone of these words is sufficient to indicate that the strings are not identical.
  - If  $I$  and  $J$  do not have at least some words in common, then we can decide that the strings are not similar.
- We know that when  $P(A) = P(C) = P(G) = P(T) = 0.25$ , the probability that an octamer beginning at any position in string  $J$  will correspond to a particular octamer in the list above is  $1/4^8$ .
  - Provided that  $J$  is short, this is not very probable.
  - If  $J$  is long, then it is quite likely that one of the eight-letter words in  $I$  can be found in  $J$  by chance.
- The appearance of a subset of these words is a *necessary but not sufficient* condition for declaring that  $I$  and  $J$  have meaningful sequence similarity.

## Word Lists and Comparison by Content

- Rather than scanning each sequence for each k-word, there is a way to collect the k-word information in a set of lists.
- A list will be a row of a table, where the table has  $4^k$  rows, each of which corresponds to one k-word.
- For example, with  $k = 2$  and the sequences below,

```
J = C C A T C G C C A T C G  
I = G C A T C G G C
```

we obtain the word lists shown in the table on the next slide (Table 7.2, Deonier et al 2005).

# Search space reduction

J		I	
AA		AA	
AC		AC	
AG		AG	
AT	3, 9	AT	3
CA	2, 8	CA	2
CC	1, 7	CC	
CG	5, 11	CG	5
CT		CT	
GA		GA	
GC	6	GC	1, 7
GG		GG	6 ←
GT		GT	
TA		TA	
TC	4, 10	TC	4
TG		TG	
TT		TT	

## Search space reduction

- Thinking of the rows as k-words, we denote the list of positions in the row corresponding to the word  $w$  as  $L_w(J)$ 
  - e.g., with  $w = CG$ ,  $L_{CG}(J) = \{5,11\}$
- These tables are sparse, since the sequences are short
- Time is money:
  - The tables can be constructed in a time proportional to the sum of the sequence lengths.
  - One approach to speeding up comparison is to limit detailed comparisons only to those sequences that share enough "content"
    - Content sharing = k-letter words in common

## Search space reduction

- The statistic that counts k-words in common is

$$\sum_{i=1}^{n-k+1} \sum_{j=1}^{m-k+1} X_{i,j},$$

where  $X_{i,j} = 1$  if  $I_i I_{i+1} \dots I_{i+k-1} = J_j J_{j+1} \dots J_{j+k-1}$  and 0 otherwise.

- The computation time is proportional to  $n \times m$ , the product of the sequence lengths.
- To improve this, note that for each  $w$  in  $I$ , there are  $\#L_w(J)$  occurrences in  $J$ . So the sum above is equal to:

$$\sum_w (\#L_w(I)) \times (\#L_w(J)).$$

- Note that this equality is a restatement of the relationship between + and x

## Search space reduction

- The second computation is much easier.
  - First we find the frequency of k-letter words in each sequence. This is accomplished by scanning each sequence (of lengths n and m).
  - Then the word frequencies are multiplied and added.
  - Therefore, the total time is proportional to  $4^k + n + m$ .
- For our sequence of numbers of 2-word matches, the statistic above is

$$0^2 + 0^2 + 0^2 + 2 \times 1 + 2 \times 1 + 2 \times 0 + 2 \times 1 + 0^2 + 0^2 + 1 \times 2 + 0 \times 1 \\ + 0^2 + 0^2 + 2 \times 1 + 0^2 + 0^2 = 10$$

- If 10 is above a threshold that we specify, then a full sequence comparison can be performed.
  - Low thresholds require more comparisons than high thresholds.

## Search space reduction

- This aforementioned method is quite fast, but the comparison totally ignores the relative positions of the k-words in the sequence.
- Can we come up with a more sensitive method?



## 6.c Binary Searches

- Suppose  $I = \text{GGAATAGCT}$ ,  $J = \text{GTACTGCTAGCCAAATGGACAATAGCTACA}$ , and we wish to find all  $k$ -word matches between the sequences with  $k = 4$ .
- In this example, we can readily find the matches by inspection, but we want to illustrate a general approach that would help with a bigger problem, say sequences that could be decomposed into word lists that were 5000 entries long with  $k$ -words having  $k = 10$ .

## Binary Searches

- Our method using  $k = 4$  depends on putting the 4-words in J into a list ordered alphabetically as in the table below (Table 7.1.; Deonier et al 2005)

---

I = GGAATAGCT

J (4-word, position in sequence)

AAAT 13	CTAG 7
AATA 21	CTGC 4
AATG 14	GACA 18
ACAA 19	GCCA 10
ACTG 3	GCTA 6, 25
AGCC 9	GGAC 17
AGCT 24	GTAC 1
ATAG 22	TACA 27
ATGG 15	TACT 2
CAAA 12	TAGC 8, 23
CAAT 20	TGGA 16
CCAA 11	TGCT 5
CTAC 26	

---

## Binary Searches

- Beginning with GGAA, we look in the J list for the 4-words contained in I by binary search.
- Since list J (of length  $m = 25$ ) is stored in a computer, we can extract the entry number  $m/2$ , which in this example is entry 13, CTAC.
- In all cases we round fractions up to the next integer.
- Then we proceed as follows:

## Binary Searches

- Step 1: Would GGAA be found before entry 13 in the alphabetically sorted list?
  - Since it would not, we don't need to look at the first half of the list.
- Step 2: In the second half of the list, would GGAA occur before the entry at position  $m/2 + m/4$ 
  - i.e., before entry (18.75 hence) 19, GGAC
  - GGAA would occur before this entry, so that after only two comparisons -we have eliminated the need to search 75% of the list and narrowed the search to one quarter of the list.
- Step 3: Would GGAA occur after entry 13 but at or before entry 16?
  - We have split the third quarter of the list into two  $m/8$  segments.
  - Since it would appear after entry 16 but at or before entry 19, we need only examine the three remaining entries.
- Steps 4 and 5: Two more similar steps are needed to conclude that GGAA is not contained in J.

## Binary Searches

- Had we gone through the whole ordered list sequentially, 19 steps would have been required to determine that the word GGAA is absent from J.
- With the binary search, we used only five steps.
- We proceed in similar fashion with the next 4-word from I, GAAT. We also fail to find this word in J, but the word at position 3 of I, AATA, is found in the list of 4-words from J, corresponding to position 21 of J.
- Words in I are taken in succession until we reach the end.
- Remark:  
With this method, multiple matchings are found. This process is analogous to finding a word in a dictionary by successively splitting the remaining pages in half until we find the page containing our word.

## Binary Searches

- In general, if we are searching a list of length  $m$  starting at the top and going item by item, on average we will need to search half the list before we find the matching word (if it is present).
- If we perform a binary search as above, we will need only  $\log_2(m)$  steps in our search.
- This is because  $m = 2^{\log_2(m)}$ , and we can think of all positions in our list of length  $m$  as having been generated by  $\log_2(m)$  doublings of an initial position.
- In the example above,  $m=30$ . Since  $32 = 2^5$ , we should find any entry after five binary steps. Note  $\log_2(30)=4.9$

## Toy example

- Suppose a dictionary has 900 pages
- Then finding the page containing the 9-letter word “crescendo” in this dictionary, using the binary search strategy, should require how many steps?
  - The list is 900 pages long
  - $2^9 = 512$
  - $2^{10} = 1024$
  - $512 < 900 < 1024$
- Within the page, you can again adopt a binary search to find the correct word among the list of words on that page

## Rare Words and Sequence Similarity

- For large word sizes  $k$ , the table size to be used in the “comparison by content” search can be enormous, and it will be mostly empty.
- For large  $k$ , another method for detecting sequence similarity is to put the  $k$ -words in an ordered list.
  - To find  $k$ -word matches between  $I$  and  $J$ , first break  $I$  down into a list of  $n - k + 1$   $k$ -words and  $J$  into a list of  $m - k + 1$   $k$ -words.
  - Then put the words in each list in order, from  $AA \dots A$  to  $TT \dots T$ .
  - For your information: This takes time  $n \log(n)$  and  $m \log(m)$  by standard methods which are routinely available but too advanced to present here.
  - Let's index the list by  $(W(i), Pw(i))$ ,  $i = 1, \dots, n - k + 1$  and  $(V(j), Pv(j))$ ,  $j = 1, \dots, m - k + 1$ , where, for example,  $W(i)$  is the  $i$ th word in the ordered list and  $Pw(i)$  is the position that word had in  $I$ .



## Rare Words and Sequence Similarity

- We discover k-word matches by the following algorithm, which merges two ordered lists into one long ordered list.
  - Start at the beginning of one list.
  - Successively compare elements in that list with elements in the second list.
    - If the element in the first list is smaller, include it in the merged list and continue.
    - If not, switch to the other list.
  - Proceed until reaching the end of one of the lists.

## Rare Words and Sequence Similarity

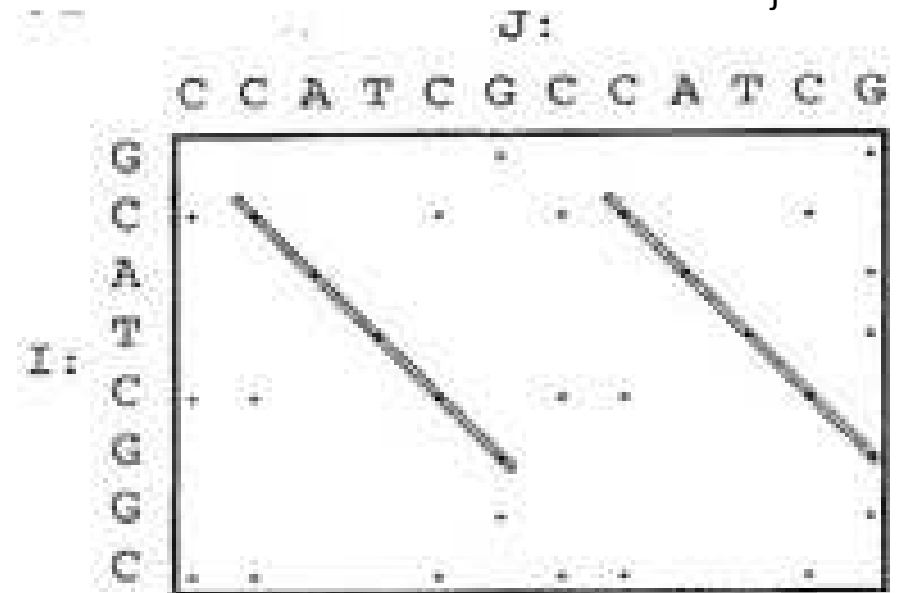
- During this process we will discover all k-words that are equal between the lists, along with producing the merged ordered list. Because the positions in the original sequences are carried along with each k-word, we will know the location of the matches as well.
- Matches longer than length k will be observed as successive overlapping matches.

## Looking for regions of similarity using FASTA

- FASTA (Pearson and Lipman, 1988) is a rapid alignment approach that combines methods to reduce the search space
- FASTA (pronounced FAST-AYE) stands for **FAST-ALL**, reflecting the fact that it can be used for a fast protein comparison or a fast nucleotide comparison.
- It depends on **k-tuples and Smith-Waterman local sequence alignment**.
  - Exercise:
    - Have you heard about the Needleman-Wunsch algorithm?
    - How is it different from a Smith-Waterman algorithm?
    - What is to be preferred and why?
- As an introduction to the rationale of the FASTA method, we begin by describing dot matrix plots, which are a very basic and simple way of visualizing regions of sequence similarity between two different strings. It allows us to identify k-tuple correspondences (*first crucial step in FASTA*)

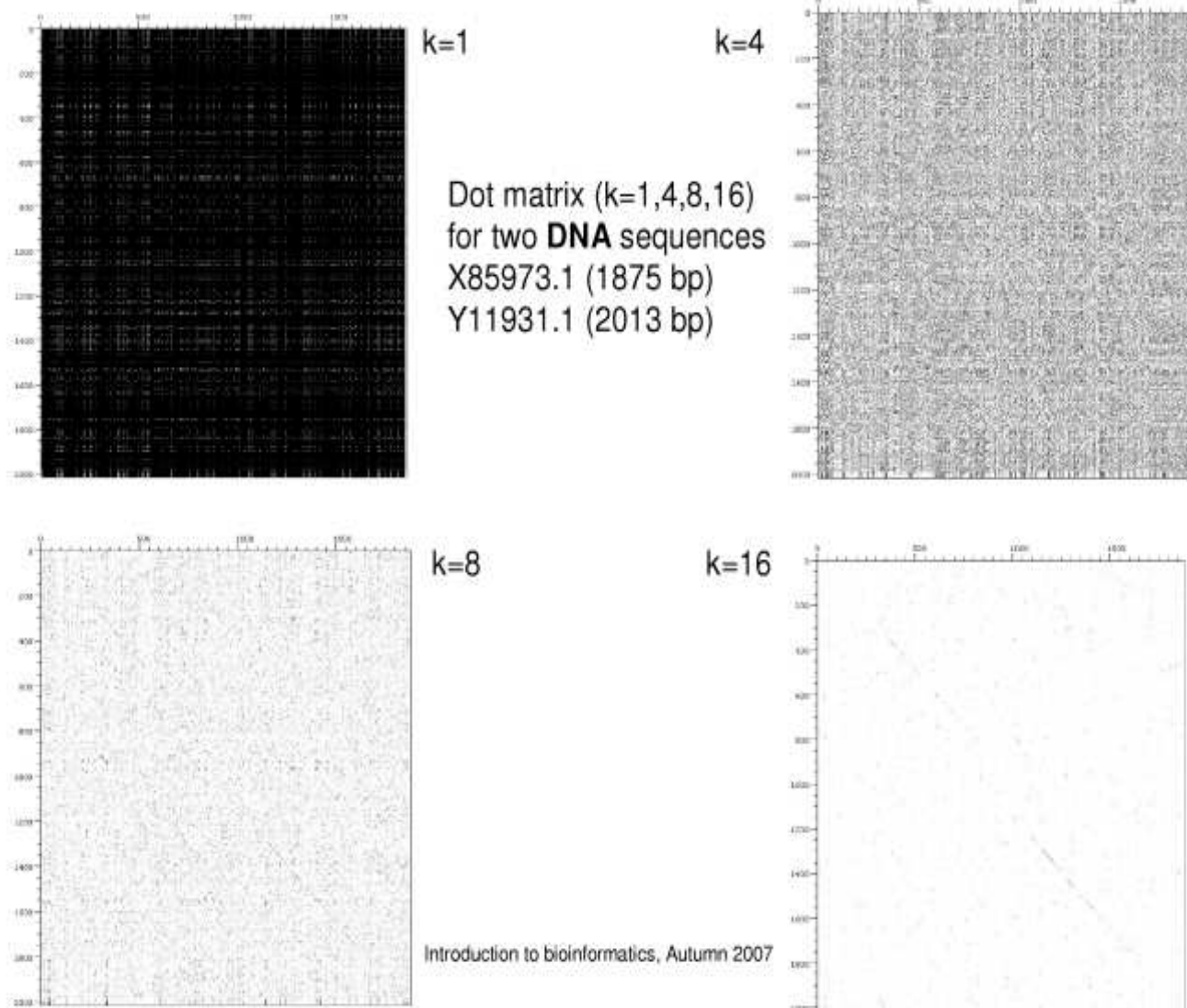
## Dot Matrix Comparisons

- Dot matrix comparisons are a special type of alignment matrix with positions  $i$  in sequence  $I$  corresponding to rows, positions  $j$  in sequence  $J$  corresponding to columns
- Moreover, sequence identities are indicated by placing a dot at matrix element  $(i,j)$  if the word or letter at  $I_i$  is identical to the word or letter at  $J_j$ .
- An example for two DNA strings is shown in the right panel. The string CATCG in  $I$  appears twice in  $J$ , and these regions of local sequence similarity appear as two diagonal arrangements of dots: diagonals represent regions having sequence similarity.



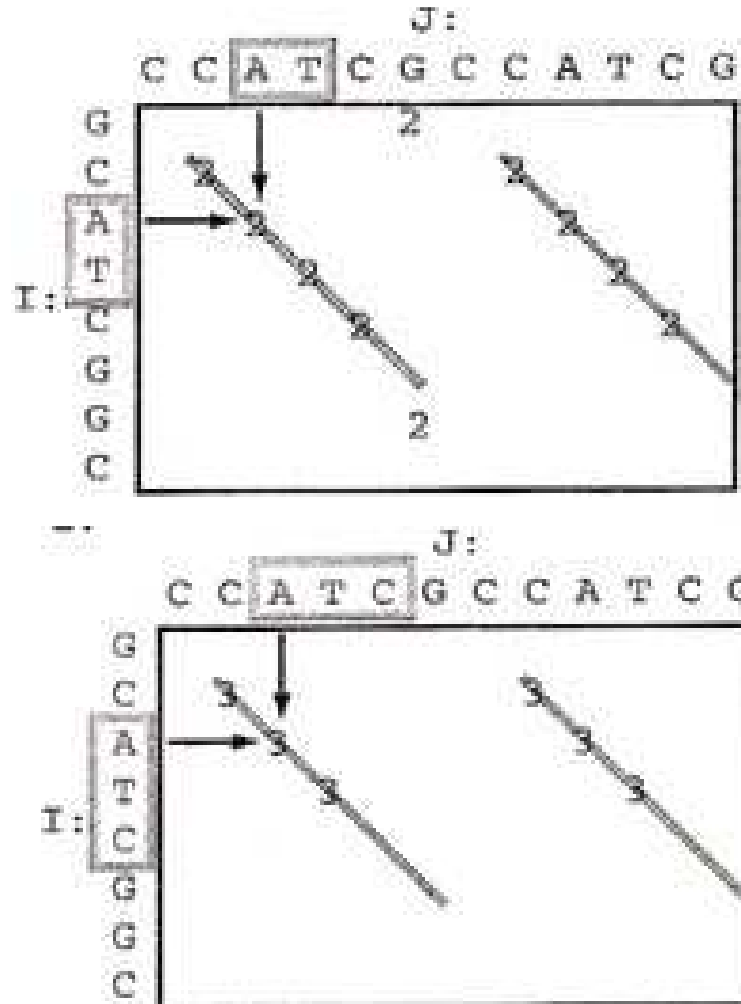
## Dot Matrix Comparisons

- When I and J are DNA sequences and are short, the patterns of this type are relatively easy to see.
- When I and J are DNA sequences and very long, there will be many dots in the matrix since, for any letter at position j in J, the probability of having a dot at any position i in I will equal the frequency of the letter  $J_j$  in the DNA.
  - For 50% A+T, this means that on average 1/4 of the matrix elements will have a dot.
- When I and J are proteins, dots in the matrix elements record matches between amino acid residues at each particular pair of positions.
  - Since there are 20 different amino acids, if the amino acid frequencies were identical, the probability of having a dot at any particular position would be 1/20.



## 6.d FASTA

- The rationale for FASTA (Wilbur and Lipman, 1983) can be visualized by considering what happens to a dot matrix plot when we record matches of  $k$ -tuples ( $k > 1$ ) instead of recording matches of single letters (Fig. 7.1; Deonier et al 2005).



## FASTA: Rationale

- We again place entries in the alignment matrix, except this time we only make entries at the first position of each dinucleotide or trinucleotide (k-tuple matches having  $k = 2$  (plotted numerals 2) or  $k = 3$  (plotted numerals 3)).
- The number of matrix entries is reduced as  $k$  increases.
- By looking for words with  $k > 1$ , we find that we can ignore most of the alignment matrix since the absence of shared words means that subsequences don't match well.
  - There is no need to examine areas of the alignment matrix where there are no word matches. Instead, we only need to focus on the areas around any diagonals.
- Our task is now to compute efficiently diagonal sums of scores,  $S_i$ , for diagonals. How can we compute these scores?



## FASTA: Rationale

- Consider again the two strings I and J that we used before:

```
J = C C A T C G C C A T C G
I = G C A T C G G C
```

- Scores can be computed in the following way:
  - Make a k-word list for J.

J		I	
AA		AA	
AC		AC	
AG		AG	
AT	3, 9	AT	3
CA	2, 8	CA	2
CC	1, 7	CC	
CG	5, 11	CG	5
CT		CT	
GA		GA	
GC	6	GC	1, 7
GG		GG	6
GT		GT	
TA		TA	
TC	4, 10	TC	4
TG		TG	
TT		TT	

## FASTA: Rationale

- Then initialize all row sums to 0:

$l$	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6
$S_l$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

(Why does  $l$  not range from -11 to 7?)

- Next proceed with the 2-words of  $I$ , beginning with  $i = 1$ , GC. Looking in the list for  $J$ , we see that  $L_{GC}(J) = \{6\}$ , so we know that at  $l = 1 - 6 = -5$  there is a 2-word match of GC.
  - Therefore, we replace  $S_{-5} = 0$  by  $S_{-5} = 0 + 1 = 1$ .
- Next, for  $i = 2$ , we have  $L_{CA}(J) = \{2, 8\}$ .
  - Therefore replace  $S_{2-2} = S_0 = 0$  by  $S_0 = 0 + 1$ , and replace  $S_{2-8} = S_{-6} = 0$  by  $S_{-6} = 0 + 1$ .

## FASTA: Rationale

- These operations, and the operations for all of the rest of the 2-words in  $I$ , are summarized below.
  - Note that for each successive step, the then-current score at  $S_i$  is employed:  $S_0$  was set to 1 in step 2, so  $l$  is incremented by 1 in step 3.

$$\begin{array}{llll}
 i = 1, \text{ GC} & L_{\text{GC}}(J) = \{6\} & l = 1 - 6 = -5 & \\
 & & S_{-5} = 0 \rightarrow S_{-5} = 0 + 1 = 1 & \\
 i = 2, \text{ CA} & L_{\text{CA}}(J) = \{2, 8\} & l = 2 - 2 = 0 & \\
 & & S_0 = 0 \rightarrow S_0 = 0 + 1, \text{ and} & \\
 & & l = 2 - 8 = -6 & \\
 & & S_{-6} = 0 \rightarrow S_{-6} = 0 + 1 & 
 \end{array}$$

## FASTA: Rationale

$i = 3, \text{AT}$	$L_{\text{AT}}(J) = \{3, 9\}$	$l = 3 - 3 = 0$ $S_0 = 1 \rightarrow S_0 = 1 + 1 = 2$ $l = 3 - 9 = -6$ $S_{-6} = 1 \rightarrow S_{-6} = 1 + 1 = 2$
$i = 4, \text{TC}$	$L_{\text{TC}}(J) = \{4, 10\}$	$l = 4 - 4 = 0$ $S_0 = 2 \rightarrow S_0 = 2 + 1 = 3$ $l = 4 - 10 = -6$ $S_{-6} = 2 \rightarrow S_{-6} = 2 + 1 = 3$
$i = 5, \text{CG}$	$L_{\text{CG}}(J) = \{5, 11\}$	$l = 5 - 5 = 0$ $S_0 = 3 \rightarrow S_0 = 3 + 1 = 4$ $l = 5 - 11 = -6$ $S_{-6} = 3 \rightarrow S_{-6} = 3 + 1 = 4$
$i = 6, \text{GG}$	$L_{\text{GG}}(J) = \{\emptyset\}$	$L_{\text{GG}}(J)$ is the empty set: no sums are increased
$i = 7, \text{GC}$	$L_{\text{GC}}(J) = \{6\}$	$l = 7 - 6 = 1$ $S_1 = 0 \rightarrow S_1 = 0 + 1 = 1$

$l$	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6
$S_l$	0	0	0	0	4	1	0	0	0	0	4	1	0	0	0	0	0

## FASTA: Rationale

- In conclusion, in our example there are  $7 \times 11 = 77$  potential 2-matches, but in reality there are ten 2-matches with four nonzero diagonal sums.

Where do 7 and 11 come from?

- We have indexed diagonals by the offset,  $l = i - j$ .
- In this notation, the nonzero diagonal sums are  $S_{+1} = 1$ ,  $S_0 = 4$ ,  $S_{-5} = 1$ , and  $S_{-6} = 4$ .
- It is possible to find these sums in time proportional to  $n + m + \#\{\text{k-word matches}\}$ .

## FASTA: rationale

- Notice that we only performed additions when there were 2-word matches.
- It is possible to find local alignments using a gap length penalty of  $-gx$  for a gap of length  $x$  along a diagonal.
  - Let  $A_i$  be the local alignment score and  $S_i$  be the maximum of all of the  $A_i$ 's on the diagonal.

## FASTA: rationale

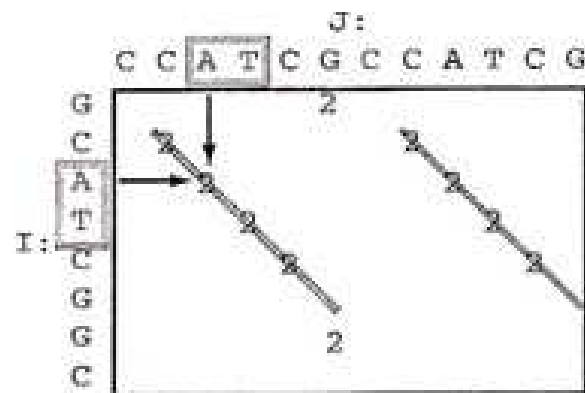
Five steps are involved in FASTA:

- Step 1: Use the look-up table to identify k-tuple identities between I and J.
- Step 2: Score diagonals containing k-tuple matches, and identify the ten best diagonals in the search space.
- Step 3: Rescore these diagonals using an appropriate scoring matrix (especially critical for proteins), and identify the subregions with the highest score (initial regions).
- Step 4: Join the initial regions with the aid of appropriate joining or gap penalties for short alignments on offset diagonals.
- Step 5: Perform dynamic programming alignment within a band surrounding the resulting alignment from step 4.

## FASTA: rationale

- Step 1: *identify k-type identities*

- To implement the first step, we pass through I once and create a table of the positions i for each possible word of predetermined size k.
- Then we pass through the search space J once, and for each k-tuple starting at successive positions j, "look up" in the table the corresponding positions for that k-tuple in I.
- Record the i,j pairs for which matches are found: the i,j pairs define where potential diagonals in the alignment matrix can be found.





## FASTA: rationale

- Step 2: *identify high-scoring diagonals*
  - If I has  $n$  letters and J has  $m$  letters, then there are  $n+m-1$  diagonals.
    - Think of starting in the upper left-hand corner, drawing successive diagonals all the way down, moving your way through the matrix from left to right ( $m$  diagonals).
    - Start drawing diagonals through all positions in I ( $n$  diagonals).
    - Since you will have counted the diagonal starting at  $(1,1)$  twice, you need to subtract 1.
  - Note that we have seen before how to score sub-diagonals along a diagonal, with or without accounting for gaps.

## FASTA: rationale

- Step 2: ***identify high-scoring diagonals***
  - To score the diagonals, calculate the number of k-tuple matches for every diagonal having at least one k-tuple (*identified in step 1*).
    - Scoring may take into account distances between matching k-tuples along the diagonal.
    - Note that the number of diagonals that needs to be scored will be much less than the number of all possible diagonals (*reduction of search space*).
  - Identify the significant diagonals as those having significantly more k-tuple matches than the mean number of k-tuple matches.
    - For example, if the mean number of 6-tuples is  $5 \pm 1$ , then with a threshold of two standard deviations, you might consider diagonals having seven or more 6-tuple matches as significant.
  - Take the top ten significant diagonals.

## FASTA: rationale

### • Step 3: *Rescore the diagonals*

- We rescore the diagonals using a scoring table to find subregions with identities shorter than k.
- Rescoring reveals sequence similarity not detected because of the arbitrary demand for uninterrupted identities of length k.
- The need for this rescoring is illustrated by the two examples below.

I: C C A T C G C C A T C G                      (Number 4-tuple matches: 0)  
 J: C C A A C G C A A T C A

I': C C A T C G C C A T C G  
 J': A C A T C A A A T A A A

- In the first case, the placement of mismatches spaced by three letters means that there are no 4-tuple matches, even though the sequences are 75% identical.
  - The second pair shows one 4-tuple match, but the two sequences are only 33% identical.
- We retain the subregions with the highest scores.

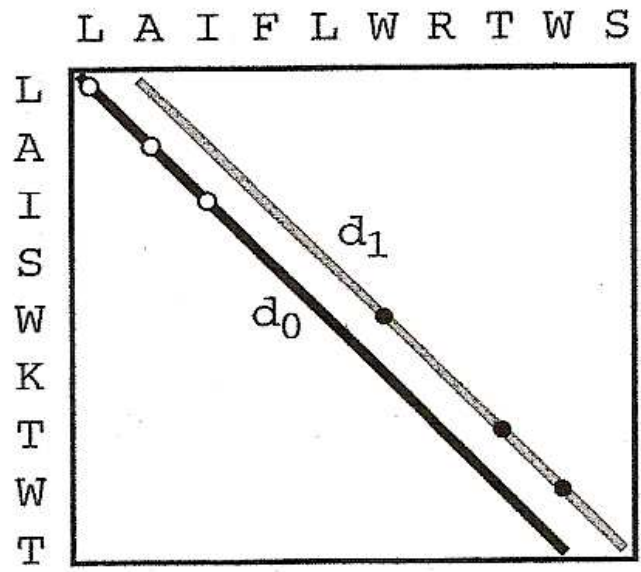
## FASTA: rationale

- Step 4: *Joining diagonals*

- Diagonals may be offset from each other, if there were a gap in the alignment (i.e., vertical or horizontal displacements in the alignment matrix, as described in the previous chapter).
- Such offsets may indicate indels, suggesting that the local alignments represented by the two diagonals should be joined to form a longer alignment.
- Diagonal  $d_i$  is the one having  $k$ -tuple matches at positions  $i$  in string  $I$  and  $j$  in string  $J$  such that  $i - j = l$ . As described before,  $l = i - j$  is called the offset.
- Alignments are extended by joining offset diagonals if the result is an extended aligned region having a higher alignment score, taking into account appropriate joining (gap) penalties.

# Offset diagonals

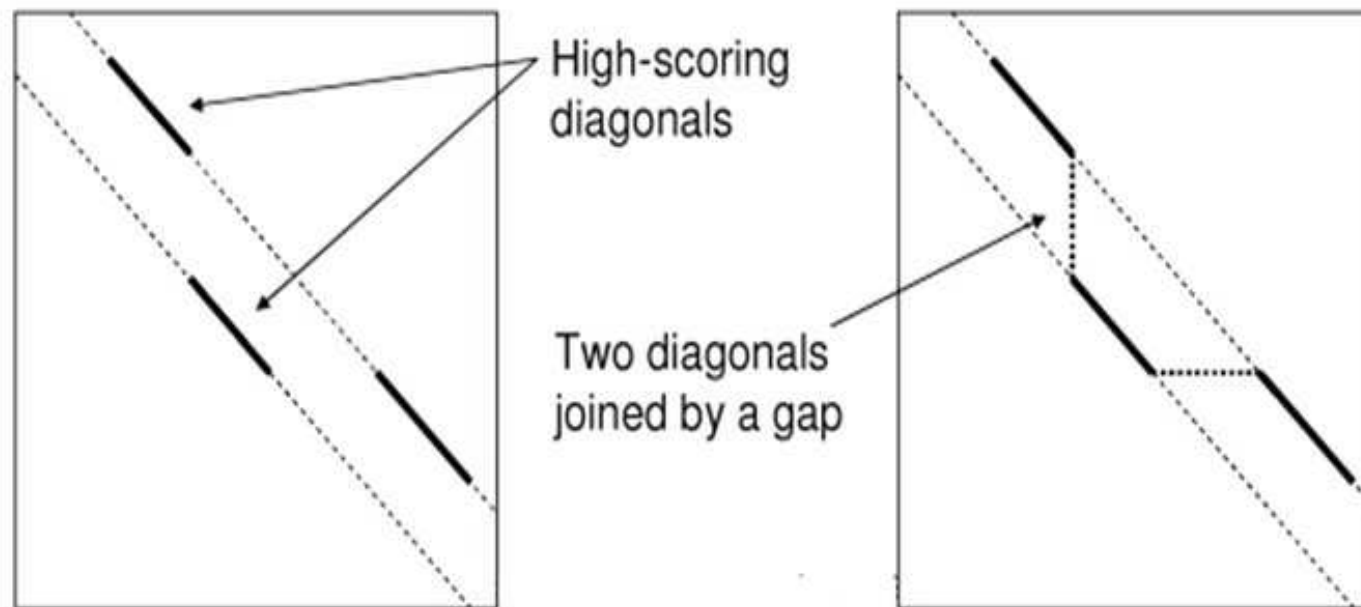
	1	2	3	4	5	6	7	8	9	10
I:	L	A	I	F	L	W	R	T	W	S
J:	L	A	I	S	W	K	T	W	T	



- j=1, i=1, i-j=0
- j=2, i=2, i-j=0
- j=3, i=3, i-j=0
  
- j=5, i=6, i-j=1
  
- j=7, i=8, i-j=1
- j=8, i=9, i-j=1

## Offset diagonals

- The idea is to find the best-scoring combination of diagonals
- Two offset diagonals can be joined with a gap, if the resulting alignment has a higher score
- Note that different gap open and extension penalties may be used



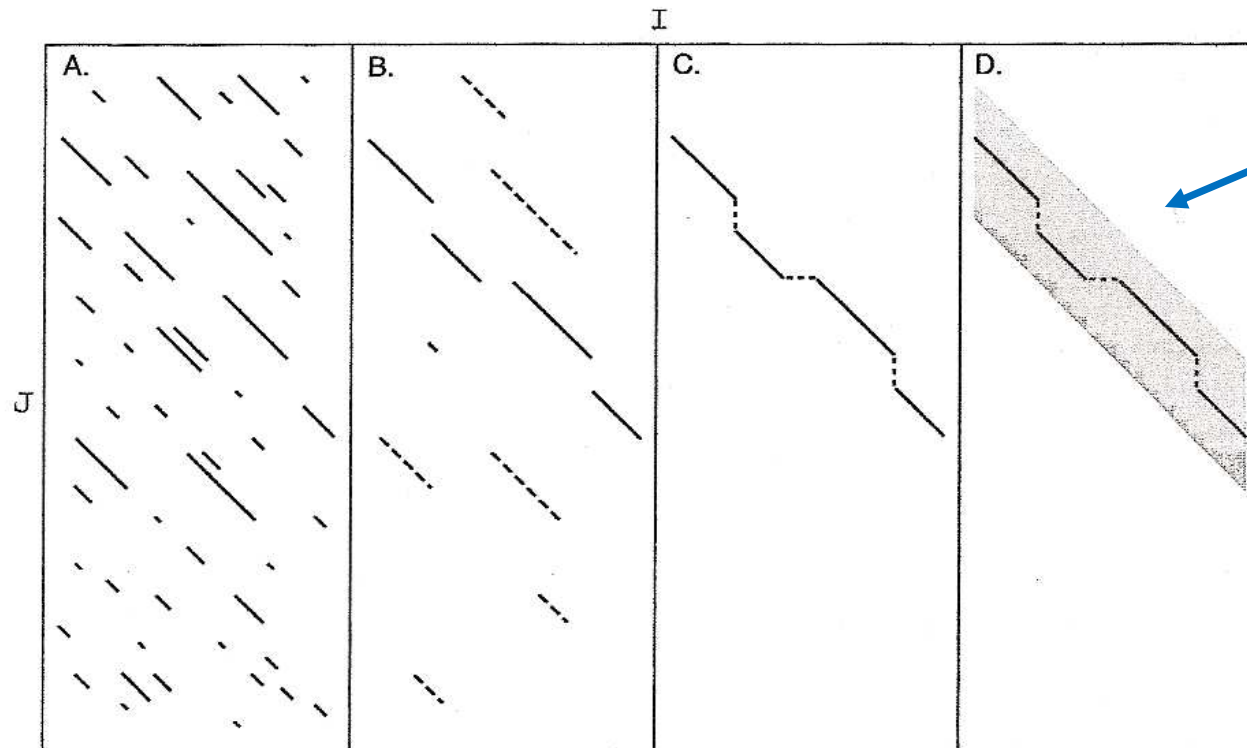
## FASTA: rationale

- Step 5: ***Smith- Waterman local alignment***

- The last step of FASTA is to perform local alignment using dynamic programming round the highest-scoring
- The region to be aligned covers  $-w$  and  $+w$  offset diagonal to the highest scoring diagonals
- With long sequences, this region is typically very small compared to the entire  $n \times m$  matrix (hence, once again, a reduction of the search space was obtained)

## FASTA: rationale

In FASTA, the alignment step can be restricted to a comparatively narrow window extending  $+w$  to the right and  $-w$  to the left of the positions included within the highest-scoring diagonal (dynamic programming)





## FASTA in practice

- To do a FASTA search, you first need to prepare an appropriate input file

```
LIB SWALL
WORD 1
LIST 50
TITLE HALHA
SEQ
PTVEYLNRYETLDDQGWDMDDDDLFEKAADAGLDGEDYGTMEVAEGEYILEAAEAQGYDWP
FSCRAGACANCASIVKEGEIDMDMQQILSDEEVEEKDVRLTCIGSPADEVKIVYNAKHL
DYLQNRVI
```

- The first line contains the data library files to be searched (in this case all Swiss-Prot and NBRF/PIR entries). It may be EMALL (all EMBL entries plus those in the latest release), or GENEMBL (GenBank plus EMBL), or EPRI (EMBL primate entries), etc.
- The second line gives the word size or k-tuple value.

## FASTA in practice

```
WORD 1
LIST 50
TITLE HALHA
SEQ
PTVEYLN YETLDDQGWDMDDDDLFEKAADAGLDGEDYGTMEVAEGEYILEAAEAQGYDWP
FSCRAGACANCASIVKEGEIDMDMQQILSDEEVEEKDVRLTCIGSPADEVKIVYNAKHL
DYLQNRVI
```

- The third line says to LIST on the output the top 50 scores.
- The TITLE line is used for the subject of the mail message.
- Finally SEQ implies that everything below this line to the end of the message is part of the sequence. In this case the sequence is the protein sequence of the ferredoxin gene of *Halobacterium halobium*.
- After creating this file, mail the file by electronic mail to [fasta@ebi.ac.uk](mailto:fasta@ebi.ac.uk) and the results will be sent back by electronic mail.

## FASTA in practice

- However, there are easier ways and more information about these can be found at several loci on the web. For instance,

<http://www.biocenter.helsinki.fi/bi/rnd/biocomp/prog5.html>

EMBL-EBI EB-eye Search All Databases: Enter Text Here Go Reset Advanced Search Give us feedback

Databases Tools EBI Groups Training Industry About Us Help Site Index

EBI > Tools > Similarity & Homology

### FASTA/SSEARCH/GLSEARCH - Protein Similarity Search

Provides sequence similarity searching against protein databases using the FASTA and SSEARCH programs. SSEARCH does a rigorous Smith-Waterman search for similarity between a query sequence and a database. GGSEARCH compares a protein or DNA sequence to a sequence database producing global-global alignment (Needleman-Wunsch). GLSEARCH compares a protein or DNA sequence to a sequence database. FASTA can be very specific when identifying long regions of low similarity especially for highly diverged sequences. You can also conduct sequence similarity searching against nucleotide databases or complete proteome/genome databases using the FASTA programs.

[Download Software](#)

PROGRAM	DATABASES	RESULTS	SEARCH TITLE	YOUR EMAIL
FASTA	Protein UniProt Knowledgebase UniProtKB/Swiss-Prot UniProt Clusters 100%	interactive	Sequence	

MATRIX	GAP OPEN	GAP EXTEND	KTUP	EXPECTATION UPPER VALUE	EXPECTATION LOWER VALUE
BLOSUM5I	-10	-2	2	10.0	default

DNA STRAND	HISTOGRAM	MOLECULE TYPE
none	no	Protein

SCORES	ALIGNMENTS	SEQUENCE RANGE	DATABASE RANGE	FILTER	STATISTICAL ESTIMATES

Similar Applications: FASTA, BLAST, GGSEARCH, GLSEARCH

# FASTA in practice

EMBL-EBI EB-eye Search All Databases Enter Text Here Go Reset Advanced Search Give us feedback

Databases Tools EBI Groups Training Industry About Us Help Site Index

Tools Index  
Protein Functional Analysis  
Proteomic Services  
Sequence Analysis  
Similarity & Homology  
Structural Analysis  
Web Services  
Miscellaneous Tools  
Downloads

**FASTA @ EBI**

FASTA (pronounced FAST-AYE) stands for **FAST-ALL**, reflecting the fact that it can be used for a fast protein comparison or a fast nucleotide comparison. This program achieves a high level of sensitivity for similarity searching at high speed. This is achieved by performing optimised searches for local alignments using a substitution matrix. The high speed of this program is achieved by using the observed pattern of word hits to identify potential matches before attempting the more time consuming optimised search. The trade-off between speed and sensitivity is controlled by the ktup parameter, which specifies the size of the word. Increasing the ktup decreases the number of background hits. Not every word hit is investigated but instead initially looks for segments containing several nearby hits.

[Download Software](#)

Below is a list of all the FASTA's available at the EBI. Please note we also provide a ['Programmatic Access to FASTA'](#).

**General FASTA Programs**

Tool	Description
<a href="#">FASTA-Protein</a> ⓘ	Sequence similarity searching against protein databases using FASTA.
<a href="#">FASTA-Nucleotide</a> ⓘ	Sequence similarity searching against nucleotide databases using FASTA.

**Specialised FASTA Programs**

FASTA-...  
FASTA-...  
FASTA-...  
FASTA-...

**FASTA Related Literature**  
Search for FASTA related literature in Medline... [more](#)

**Search the UniProt Protein Resource with FASTA**  

- ▶ [UniProt](#)
- ▶ [UniParc](#)
- ▶ [UniRef100](#)
- ▶ [UniRef90](#)
- ▶ [UniRef50](#)

# FASTA in practice

EMBL-EBI

Databases
Tools
EBI Groups
Training
Industry
About Us
Help
Site Index

- Help Index
- General Help
- Formats
- Gaps
- Matrix
- References
- FASTA Help
- MView Help
- VisualFASTA Help

---

- View all FASTA's at EBI
- FASTA Programmatic Access

---

- Database Information
  - EMBL-Bank
  - IMGT/HLA

---

- Similar Applications
  - FASTA
  - BLAST
  - GGSEARCH
  - GLSEARCH

EBI > Tools > Similarity & Homology > FASTA

### FASTA - Nucleotide Similarity Search

Provides sequence similarity searching against nucleotide and protein databases using the FASTA programs. FASTA can be very specific when identifying long regions of low similarity especially for highly diverged sequences. You can also conduct sequence similarity searching against complete [proteome](#) or [genome](#) databases using the [FASTA programs](#).

[Download Software](#)

PROGRAM	DATABASES		RESULTS	SEARCH TITLE	YOUR EMAIL
FASTA	Nucleic Acid <span style="background-color: #008080; color: white; padding: 2px;">EMBL Release</span> EMBL Updates EMBL Coding Sequence		email	Sequence	
MATRIX	GAP OPEN	GAP EXTEND	KTUP	EXPECTATION UPPER VALUE	EXPECTATION LOWER VALUE
none	-14	-4	6	10.0	default
DNA STRAND	HISTOGRAM	MOLECULE TYPE			
both	no	DNA			
SCORES	ALIGNMENTS	SEQUENCE RANGE	DATABASE RANGE	FILTER	STATISTICAL ESTIMATES
50	50	START-END	START-END	none	Regress

K Van Steen

479

## FASTA in practice

http://www.ebi.ac.uk/Tools/fasta33/help.html#expup

EMBL-EBI

95	670	272:*	:*****
100	611	210:*	:*****
102	627	163:*	:*****
104	369	126:*	:*****
106	224	97:*	:*****
108	193	75:*	:*****
110	136	58:*	:*****
112	86	45:*	:*****
114	67	35:*	:*****
116	52	27:*	:*****
118	41	21:*	:*****
>120	360	16:*	:*****

- EXPECTATION VALUE UPPER LIMIT**

Here you may set the expectation value upper limit for score and alignment display. Generally, in evaluating the E() scores, the following rules of thumb can be used, sequences with E() less than 0.01 are almost always found to be homologous, sequences with E() between 1 and 10 frequently turn out to be related as well.

*The defaults are 10.0 for FASTA with protein searches, 5.0 for translated DNA/protein comparisons, and 2.0 for DNA/DNA searches.*
- EXPECTATION VALUE LOWER LIMIT**

Expectation value lower limit for score and alignment display. A value of 1e-6 prevents library sequences with E()- values lower than 1e-6 from being displayed. This allows the use to focus on more distant relationships. Thus with this option if set will filter out the best matches and allow more distant relationships to be displayed.

*The default setting for this is zero.*
- SEQUENCE RANGE**

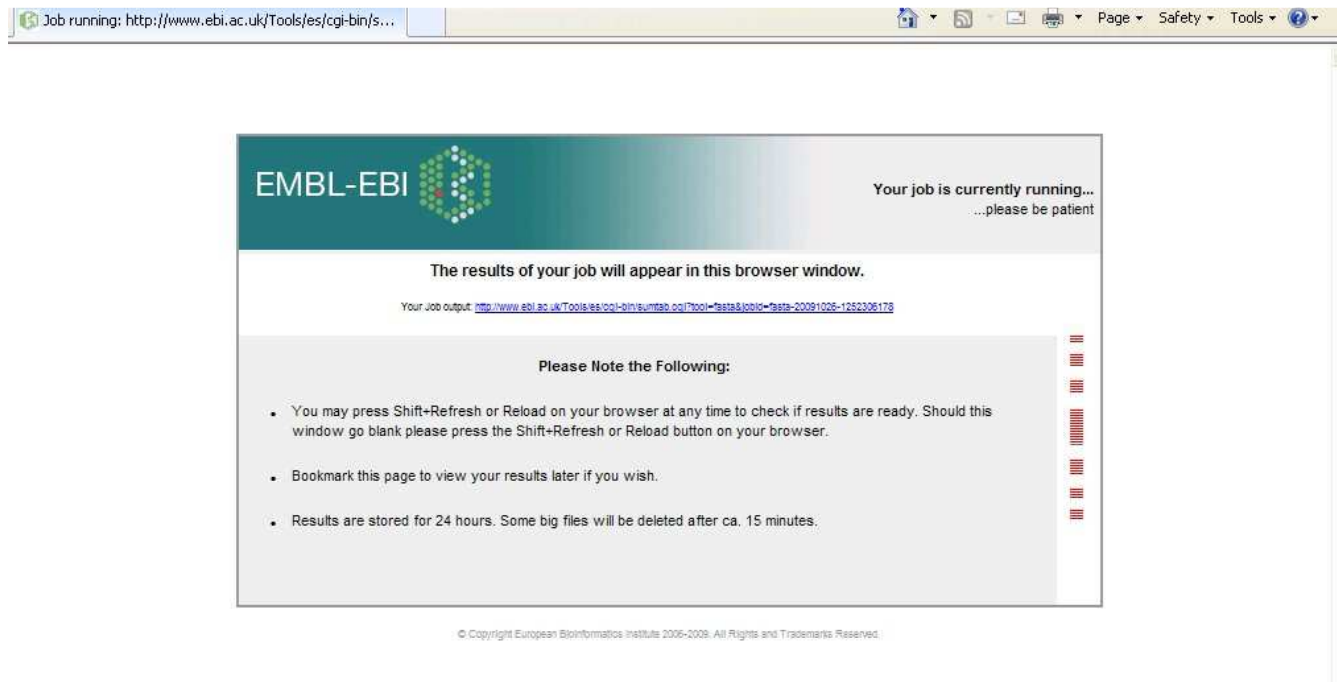
This options allows the user to denote which region within the query sequence should be searched. For example, if you submit a sequence of 380 amino acids/nucleotides, you may wish to search the database using the region comprising positions 50 to 200. In this case, the user should type into the text field the numbers: 50-200.

*The default is to search using the whole query sequence.*
- DATABASE SEQUENCE RANGE TO SEARCH**

[Top of page](#)  
[Your Sequences](#)  
[Your Email](#)  
[Search Title](#)  
[Results](#)  
[Program](#)  
[Databases](#)  
[Matrix](#)  
[Gap Penalties](#)  
[Scores](#)  
[Alignments](#)  
[KTUP](#)  
[Strand](#)  
[Histogram](#)  
[Expectation Value Upper Limit](#)  
[Expectation Value Lower Limit](#)  
[Sequence Range](#)  
[Database Sequence Range to Search](#)  
[Molecule Type](#)  
[Filter](#)  
[Sequence Input Window](#)  
[Upload a File](#)  
[References](#)  
[Example](#)  
[MView Help](#)  
[VisualFASTA Help](#)

## FASTA in practice

- When submitting a FASTA job, the following screen appears



- Because the interactive mode was selected, results will appear in the active browser

# FASTA in practice

Result summary fasta-20091026-1252306178

EMBL-EBI EB-eye Search All Databases Enter Text Here Go Reset Advanced Search Give us feedback

Databases Tools EBI Groups Training Industry About Us Help Site Index

- Help
- General Help
- Formats
- Gaps
- Matrix
- References
- Fasta Help
- View Help
- VisualFasta Help
- Database Information
- UniProt
- UniParc

### FASTA Results

SUBMISSION PARAMETERS			
Title	Sequence	Database	em_rel
Sequence length	700	Sequence type	n
Program	fasta	Version	35.04 Aug. 19, 2009
Expectation upper value	10.0	Sequence range	1-
Number of scores	50	Number of alignments	50
Word size	6	Open gap penalty	-14
Gap extension penalty	-4	Histogram	false

Alignment	DB:ID	Source	Length	Identity%	Similar%	Overlap	E()
1 <input type="checkbox"/>	<a href="#">EM_HUM:AC007728</a>	Homo sapiens chromosome 16 c	168271	100.0	100.0	700	3.2e-184
2 <input type="checkbox"/>	<a href="#">EM_HUM:AC007608</a>	Homo sapiens chromosome 16 c	169773	100.0	100.0	700	3.2e-184
3 <input type="checkbox"/>	<a href="#">EM_PAT:GC699706</a>	Sequence 14951 from patent U	39936	100.0	100.0	700	4.1e-184
4 <input type="checkbox"/>	<a href="#">EM_HUM:AJ303140</a>	Homo sapiens NOD2 gene for L	163319	99.9	99.9	700	1e-183





## FASTA in practice

- Help on nucleotide searches:  
<http://www.ebi.ac.uk/2can/tutorials/nucleotide/fasta.html>
- Help on interpretation of such search results:  
<http://www.ebi.ac.uk/2can/tutorials/nucleotide/fasta1.html>

## FASTA in practice

EMBL-EBI **EB-eye Search** All Databases

Databases Tools EBI Groups Training Industry About Us Help Site Index

- 2Can Home
- What is Bioinformatics?
- Basic Biology
- Genes & Disease
- Bioinformatics Databases
- Tutorials Index
- Nucleotide Analysis**
  - Introduction
  - Checking for vector contamination
  - BLAST similarity search
  - FASTA similarity search
  - Running a whole genome search
  - Pairwise local/global alignment
  - ClustalW2 multiple sequence alignment
  - Translate DNA/RNA into protein
- Protein and Proteomic

EBI > 2Can > Tutorials > Nucleotide Analysis

### 2Can Support Portal - Nucleotide Analysis

FASTA similarity search <<< 2/8 >>>

#### Results of FASTA search

These are the results, received via email of the FASTA search described on the previous page, have you decided what sequence 4 is? If you look at the score list, the entry in the database with the [best score](#) (In EMBL, not PATENTS) is Human beta2-syntrophin (SNT B2) mRNA. You can see from the [alignment](#) of sequence 4 with Human beta2-syntrophin (SNT B2) mRNA that these sequences are identical, so this is the same sequence.

```

FASTA searches a protein or DNA sequence data bank
version 3.4t25 Sept 2, 2005
Please cite:
W.R. Pearson & D.J. Lipman PNAS (1988) 85:2444-2448

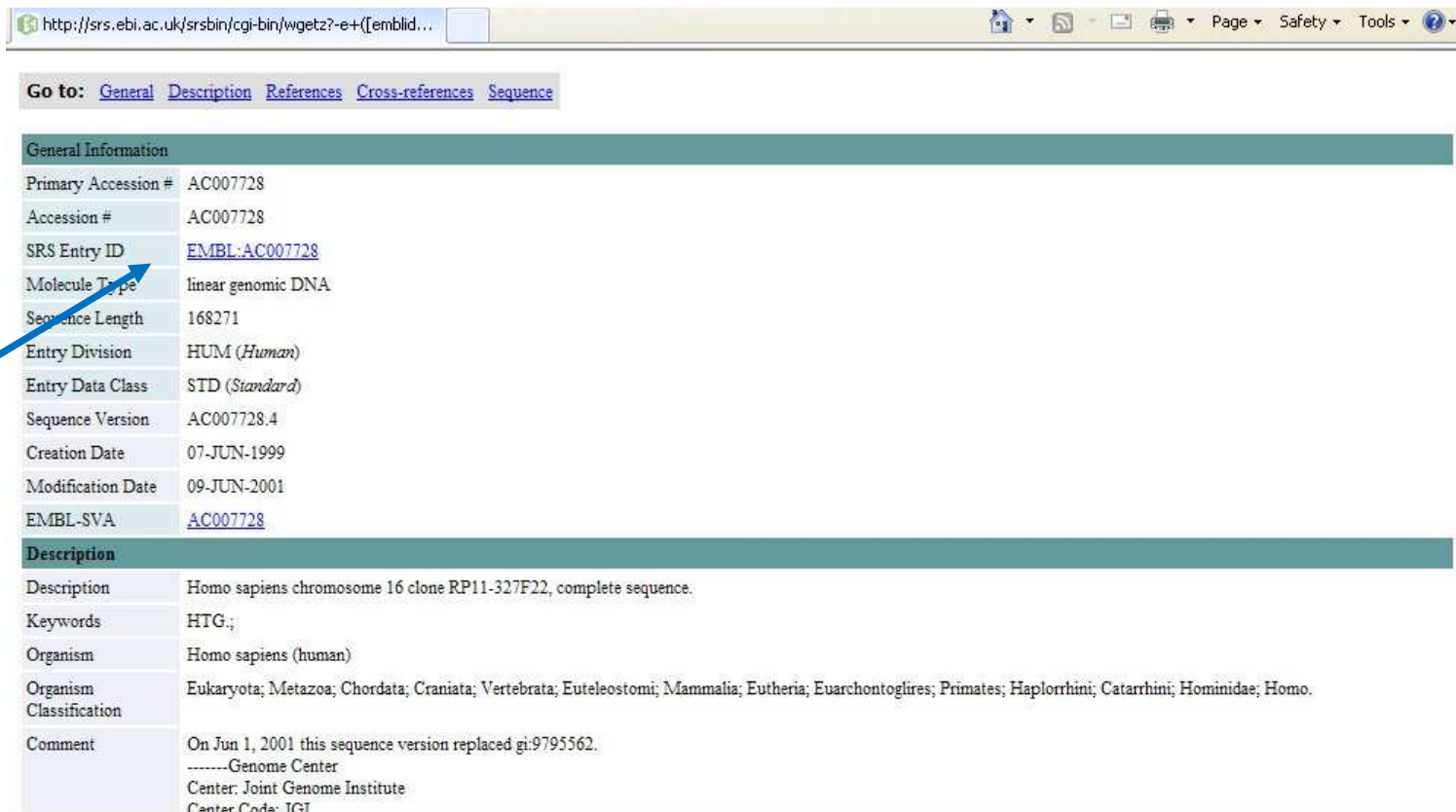
Query library @ vs +embl library
searching /ebi/services/idata/v1753/fastadb/em_fun library

1>>>_Sequence - 1700 nt
vs EMBL Nucleotide Sequence Database library

19345946543 residues in 7547515 sequences
statistics sampled from 60000 to 7670844 sequences
  
```

## FASTA in practice

- The best alignments reduce to a smaller set when we restrict attention to EMBL searches



http://srs.ebi.ac.uk/srsbin/cgi-bin/wgetz?-e+([emblid...]

Go to: [General](#) [Description](#) [References](#) [Cross-references](#) [Sequence](#)

**General Information**

Primary Accession #	AC007728
Accession #	AC007728
SRS Entry ID	<a href="#">EMBL:AC007728</a>
Molecule Type	linear genomic DNA
Sequence Length	168271
Entry Division	HUM ( <i>Human</i> )
Entry Data Class	STD ( <i>Standard</i> )
Sequence Version	AC007728.4
Creation Date	07-JUN-1999
Modification Date	09-JUN-2001
EMBL-SVA	<a href="#">AC007728</a>

**Description**

Description	Homo sapiens chromosome 16 clone RP11-327F22, complete sequence.
Keywords	HTG.;
Organism	Homo sapiens (human)
Organism Classification	Eukaryota; Metazoa; Chordata; Craniata; Vertebrata; Euteleostomi; Mammalia; Eutheria; Euarchontoglires; Primates; Haplorrhini; Catarrhini; Hominidae; Homo.
Comment	On Jun 1, 2001 this sequence version replaced gi:9795562. -----Genome Center Center: Joint Genome Institute Center Code: JGI

# FASTA in practice

- Look at the alignment itself ...

The screenshot shows the EMBL-EBI FASTA Alignment Display page. At the top, there is a search bar with 'All Databases' selected and a search input field containing 'Enter Text Here'. Navigation tabs include Databases, Tools, EBI Groups, Training, Industry, About Us, and Help. A sidebar on the left lists various help topics like General Help, Formats, Gaps, Matrix, References, Fasta Help, MView Help, VisualFasta Help, Database Information, UniProt, and UniParc.

The main content area is titled 'FASTA Alignment Display' and features a table of 'SUBMISSION PARAMETERS':

SUBMISSION PARAMETERS			
Title	Sequence	Database	em_rel
Sequence length	700	Sequence type	n
Program	fasta	Version	35.04 Aug. 19, 2009
Expectation upper value	10.0	Sequence range	1-
Number of scores	50	Number of alignments	50
Word size	6	Open gap penalty	-14
Gap extension penalty	-4	Histogram	false

Below the table are several buttons: Show Annotation, Summary Table, FASTA Result, MView, VisualFasta, XML, and SUBMIT ANOTHER JOB.

The FASTA alignment output is displayed in a text area:

```
>>EM_HUM:AC007728; AC007728 Homo sapiens chromosome 16 c (168271 nt)
rev-comp initn: 3500 init1: 3500 opt: 3500 Z-score: 3444.3 bits: 655.6 E(): 3.2e-184
banded Smith-Waterman score: 3500; 100.0% identity (100.0% similar) in 700 nt overlap (700-1:153168-153867)

              700      690      680
Seque-                AGCAAAAGGAGAAAGCTCCAATCTCTCAGA
                   ::::::::::::::::::::::::::::::
EM_HUM IGTATCCGTCTCATACGGTAGCCAAGGTAAGCAAAAGGAGAAAGCTCCAATCTCTCAGA
      153140  153150  153160  153170  153180  153190
```

## Properties of FASTA

- Fast compared to local alignment only using dynamic programming as such
  - A narrow region of the full alignment matrix is aligned
  - With long sequences, this region is typically very small compared to the whole matrix
- Increasing the parameter  $k$  (word length), decreases the number of hits
  - It increases specificity
  - It decreases sensitivity
- FASTA can be very specific when identifying long regions of low similarity
  - Specific method does not produce many incorrect results
  - Sensitive method produces many of the correct results

More info at <http://www.ebi.ac.uk/fasta>

(parameter  $ktup$  in the software corresponds to the parameter  $k$  in the class notes)

## Sensitivity and specificity reminder

- These concepts come from the world of “clinical test assessment”
  - TP = true positives; FP = false positives
  - TN = true negatives; FN = false negatives
- Sensitivity =  $TP / (TP + FN)$
- Specificity =  $TN / (TN + FP)$

	<b>Patients with disease</b>	<b>Patients without disease</b>
<b>Test is positive</b>	TP	FP
<b>Test is negative</b>	FN	TN

## The Smith-Waterman algorithm (local alignment)

- Needleman and Wunsch (1970) were the first to introduce a heuristic alignment algorithm for calculating homology between sequences (global alignment).
- Later, a number of variations have been suggested, among others Sellers (1974) getting closer to fulfill the requests of biology by measuring the metric distance between sequences [Smith and Waterman, 1981].
- Further development of this led to the Smith-Waterman algorithm based on calculation of local alignments instead of global alignments of the sequences and allowing a consideration of deletions and insertions of arbitrary length.



## The Smith-Waterman algorithm

- The Smith-Waterman algorithm uses individual pair-wise comparisons between characters as:

$$M_{i,j} = \max \left\{ \begin{array}{l} M_{i-1,j-1} + s(a_i, b_i) \\ M_{i-1,j} - \delta \\ M_{i,j-1} - \delta \\ 0 \end{array} \right\}.$$

Do you recognize this formula?

- The Smith-Waterman algorithm is the most accurate algorithm when it comes to search databases for sequence homology but it is also the most time consuming, thus there has been a lot of development and suggestions for optimizations and less time-consuming models. One example is BLAST [Shpaer et al., 1996].

## 6.e BLAST

- The most used database search programs are BLAST and its descendants.
- BLAST is modestly named for Basic Local Alignment Search Tool, and it was introduced in 1990 (Altschul et al., 1990).
- Whereas FASTA speeds up the search by filtering the k-word matches, BLAST employs a quite different strategy (see later).
- The net result is high-scoring local alignments, which are called "high scoring segment pairs" or HSPs.
- Hence, the output of BLAST is a list of HSPs together with a measure of the probability that such matches would occur by chance.

## BLAST rationale

In particular, three steps are involved in BLAST:

- Step 1: Find local alignments between the query sequence and a data base sequence (“seed hits”)
- Step 2: Extend the seed hits into high-scoring local alignments
- Step 3: Calculate p-values and a rank ordering of the local alignments

## Anatomy of BLAST: Finding Local Matches

- First, the query sequence is used as a template to construct a set of sub-sequences of length  $w$  that can score at least  $T$  when compared with the query (step 1).
- A substitution matrix, containing neighborhood sequences, is used in the comparison.
- Then the database is searched for each of these neighborhood sequences.
  - This can be done very rapidly because the search is for an exact match, just as our word processor performs exact searches.
  - We have not developed such sophisticated tools here, but such a search can be performed in time proportional to the sum of the lengths of the sequence and the database.

## Anatomy of BLAST: Finding Local Matches

- Let's return to the idea of using the query sequence to generate the neighborhood sequences. We will employ the same query sequence I and search space J that we used previously:

$$\begin{aligned} J &= \text{C C A T C G C C A T C G} \\ I &= \text{G C A T C G G C} \end{aligned}$$

- We use subsequences of length  $k = 5$ . For the neighborhood size, we use all 1-mismatch sequences, which would result from scoring matches 1, mismatches 0, and the test value (threshold)  $T = 4$ .
- For sequences of length  $k = 5$  in the neighborhood of GCATC with  $T = 4$  (excluding exact matches), we have:

$$\left\{ \begin{array}{c} \text{A} \\ \text{C} \\ \text{T} \end{array} \right\} \text{CATC}, \quad \text{G} \left\{ \begin{array}{c} \text{A} \\ \text{G} \\ \text{T} \end{array} \right\} \text{ATC}, \quad \text{GC} \left\{ \begin{array}{c} \text{C} \\ \text{G} \\ \text{T} \end{array} \right\} \text{TC}, \quad \text{GCA} \left\{ \begin{array}{c} \text{A} \\ \text{C} \\ \text{G} \end{array} \right\} \text{C}, \quad \text{GCAT} \left\{ \begin{array}{c} \text{A} \\ \text{G} \\ \text{T} \end{array} \right\}$$

## Anatomy of BLAST: Finding Local Matches

$$\left\{ \begin{array}{c} A \\ C \\ T \end{array} \right\} \text{CATC}, \quad G \left\{ \begin{array}{c} A \\ G \\ T \end{array} \right\} \text{ATC}, \quad GC \left\{ \begin{array}{c} C \\ G \\ T \end{array} \right\} \text{TC}, \quad GCA \left\{ \begin{array}{c} A \\ C \\ G \end{array} \right\} C, \quad GCAT \left\{ \begin{array}{c} A \\ G \\ T \end{array} \right\}$$

- Each of these terms represents three sequences, so that in total there are  $1 + (3 \times 5) = 16$  exact matches to search for in J.
- For the three other 5-word patterns in I (CATCG, ATCGG, and TCGGC), there are also 16 exact 5-words, for a total of  $4 \times 16 = 64$  5-word patterns to locate in J.
- A hit is defined as an instance in the search space (database) of a k-word match, within threshold T, of a k-word in the query sequence.

## Anatomy of BLAST: Finding Local Matches

- In our example, there are several hits in I to sequence J. They are

5-words in I	5-words in J	J position(s)	Score
CATCG	CATCG	2, 8	5
GCATC	CCATC	1	4
ATCGG	ATCGC	3	4
TCGGC	TCGCC	4	4

- In actual practice, the hits correspond to a tiny fraction of the entire search space.

## Anatomy of BLAST: Finding Local Matches

- The next step (step 2) is to extend the alignment starting from these "seed" hits.
  - Starting from any seed hit, this extension includes successive positions, with corresponding increments to the alignment score.
  - This is continued until the alignment score falls below the maximum score attained up to that point by a specified amount.
- Later, improved versions of BLAST only examine diagonals having two non-overlapping hits no more than a distance  $A$  residues away from each other, and then extend the alignment along those diagonals.
- Unlike the earlier version of BLAST, gaps can be accommodated in the later versions.



## Anatomy of BLAST: Finding Local Matches

- With the original version of BLAST, over 90% of the computation time was employed in producing the un-gapped extensions from the hits.
  - This is because the initial step of identifying the seed hits was effective in making this alignment tool very fast.
- Later versions of BLAST require the same amount of time to find the seed hits and have reduced the time required for the un-gapped extensions considerably.
- Even with the additional capabilities for allowing gaps in the alignment, the newer versions of BLAST run about three times faster than the original version (Altschul et al, 1997).

## Anatomy of BLAST: Scores

- Another aspect of a BLAST analysis is to rank-order by p-values the sequences found (step 3).
- If the database is  $D$  and a sequence  $X$  scores  $S(D,X) = s$  against the database, the p-value is  $P(S(D,Y) \geq s)$ , where  $Y$  is a random sequence.
- The smaller the p-value, the greater the "surprise" and hence the greater the belief that something real has been discovered.
- A p-value of 0.1 means that with a collection of query sequences picked at random, in 1/10 of the instances a score that is as large or larger would be discovered.
- A p-value of  $10^{-6}$  means that only once in a million instances would a score of that size appear by chance alone.
- There is a nice way of computing BLAST p-values that has a solid mathematical basis.

## Anatomy of BLAST: p-value derivation – intuitive approach

- In a sequence-matching problem where the score is 1 for identical letters and  $-\infty$  otherwise (i.e., no mismatches and no indels), the best local alignment score is equal to the longest exact matching between the sequences.
- In our  $n \times m$  alignment matrix, there are (approximately)  $n \times m$  places to begin an alignment.
- Generally, an optimal alignment begins with a mismatch, and we are interested in those that extend at least  $t$  matching (identical) letters.
- Set

$$p = \mathbb{P}(\text{two random letters are equal}).$$

- The event of a mismatch followed by  $t$  identities has probability  $(1 - p)p^t$ .

## Anatomy of BLAST: p-value derivation – intuitive approach

- There are  $n \times m$  places to begin this event, so the mean or expected number of local alignments of at least length  $t$  is  $nm(1 - p)p^t$ . Obviously, we want this to be a rare event that is well-modelled by the Poisson distribution with mean:

$$\lambda = nm(1 - p)p^t,$$

$$\begin{aligned}\mathbb{P}(\text{there is local alignment } t \text{ or longer}) &\approx 1 - \mathbb{P}(\text{no such event}) \\ &= 1 - e^{-\lambda} \\ &= 1 - \exp(-nm(1 - p)p^t).\end{aligned}$$

- Recall: For Poisson distribution, we need to use the function of  $j$ :  $\frac{\lambda^j}{j!} e^{-\lambda}$  (the expected nr of occurrences is lambda)

## Anatomy of BLAST: p-value derivation – intuitive approach

- This equation is of the same form used in BLAST, which estimates

$$\mathbb{P}(S(\mathcal{D}, Y) \geq s) \approx 1 - \exp(-nm\gamma\xi^t),$$

where  $\gamma > 0$  and  $0 < \varepsilon < 1$ .

- There are conditions for the validity of this formula, in which  $\gamma$  and  $\varepsilon$  are estimated parameters, but this is the idea! (In BLAST output, the last quantity is called an E-value.)
- The take-home message of this discussion is that the probability of finding a HSP (High Scoring Segment Pair) by chance using a random query sequence  $Y$  in database  $D$  is approximately equal to the E-value.

## E-values and p-values

- *P-value; probability value:*  
this is the probability that a hit would attain at least the given score, by random chance for the search database
- *E value; expectation value:*  
this is the expected number of hits of at least the given score that you would expect by random chance for the search database
- E-values are easier to interpret than p-values  
If the E-value is small enough, than it is essentially a p-value (say  $E < 0.10$ )

## E-values and p-values

- E-value  $< 10e-100$ : Identical sequences.
  - You will get long alignments across the entire query and hit sequence.
- $10e-50 < \text{E-value} < 10e-100$ : Almost identical sequences.
  - A long stretch of the query protein is matched to the database.
- $10e-10 < \text{E-value} < 10e-50$ : Closely related sequences.
  - Could be a domain match or similar.
- $1 < \text{E-value} < 10e-6$ : Could be a true homologue but it is a gray area.

## BLAST in practice

- A sequence in FASTA format begins with a single-line description, followed by lines of sequence data. The description line (define) is distinguished from the sequence data by a greater-than (" $>$ ") symbol at the beginning. It is recommended that all lines of text be shorter than 80 characters in length. An example sequence in FASTA format is:

```
>gi|129295|sp|P01013|OVAX_CHICK GENE X PROTEIN (OVALBUMIN-RELATED)
QIKDLLVSSSTDLDTTLVLVNAIYFKGMWKTAFAEDTREMPPHVTKQESKPVQMMCMNNSFNVATLPAE
KMKILELPPFASGDL SMLVLLPDEVSDLERIEKTINFEKLTWNTNPNTMEKRRVKVYLPQMKIEEKYNLTS
VLMALGMTDLFIP SANLTGISSAESLKISQAVHGAFMELSEDGIEMAGSTGVIEDIKHSPESQFRADHP
FLFLIKHNPTNTIVYFGRYWSP
```

- Blank lines are not allowed in the middle of FASTA input.
- To know more about other allowable formats, please go to the NCBI BLAST website

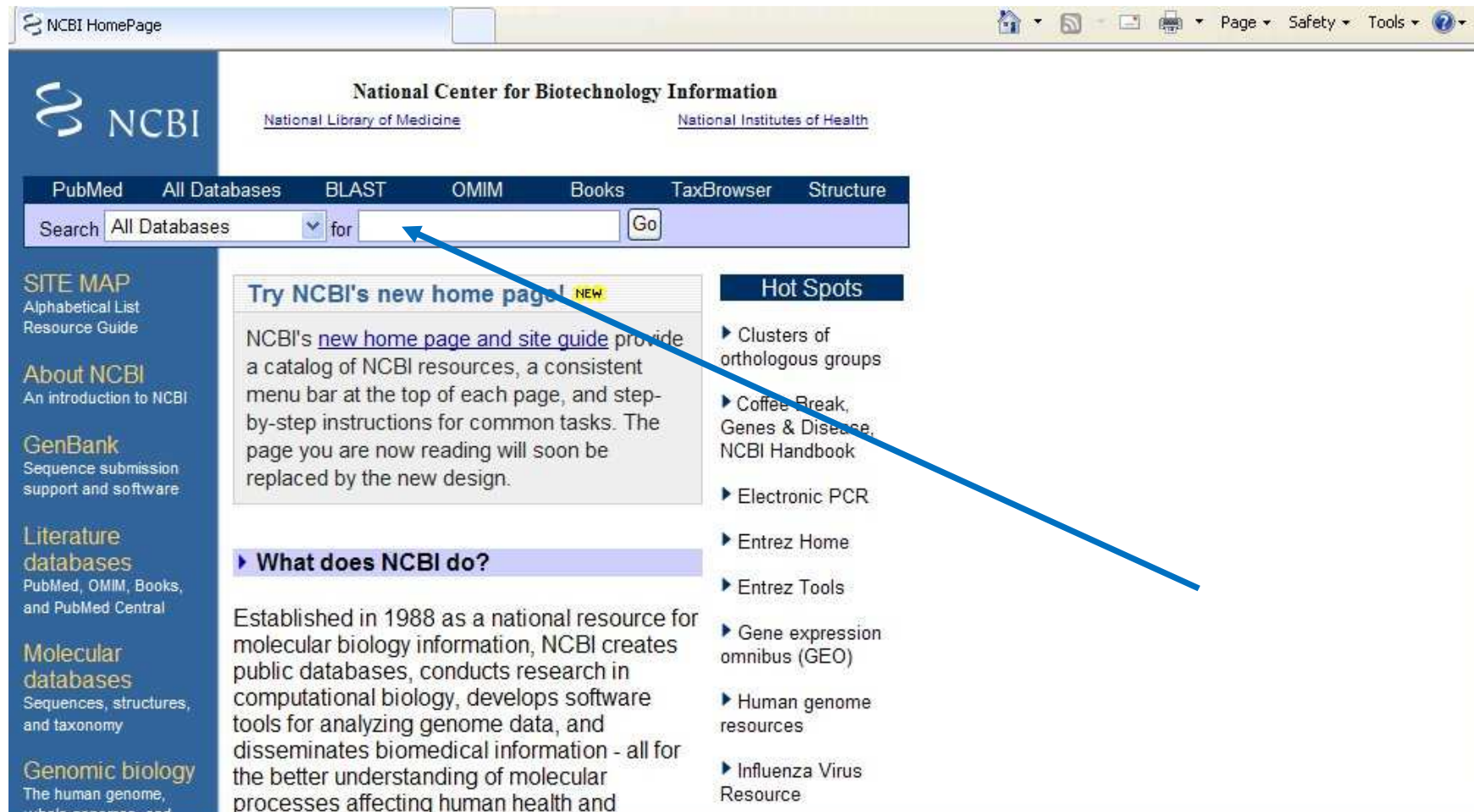


## BLAST in practice

The screenshot shows the NCBI BLAST website. At the top, there is a navigation bar with 'Home', 'Recent Results', 'Saved Strategies', and 'Help'. A 'My NCBI' section includes 'Sign In' and 'Register' links. The main content area is titled 'NCBI/ BLAST Home' and contains a description: 'BLAST finds regions of similarity between biological sequences.' Below this is a 'New' banner for 'Aligning Multiple Protein Sequences? Try the COBALT Multiple Alignment Tool.' The 'BLAST Assembled Genomes' section lists various species genomes with checkboxes: Human, Mouse, Rat, Arabidopsis thaliana, Oryza sativa, Bos taurus, Danio rerio, Drosophila melanogaster, Gallus gallus, Pan troglodytes, Microbes, and Apis mellifera. The 'Basic BLAST' section provides a list of search options: nucleotide blast, protein blast, blastx, tblastn, and tblastx, each with a brief description and the algorithms used. The 'Specialized BLAST' section is partially visible at the bottom. On the right side, there are two sidebar sections: 'News' with a link to 'BLAST 2.2.22 now available' and 'Tip of the Day' with a link to 'Use Genomic BLAST to see the genomic context'.

(<http://blast.ncbi.nlm.nih.gov/Blast.cgi>)

## BLAST in practice



The screenshot shows the NCBI homepage with a search bar at the top. A blue arrow points to the search input field. The search bar contains the text "Search All Databases" and a "Go" button. The search bar is part of a navigation menu that includes "PubMed", "All Databases", "BLAST", "OMIM", "Books", "TaxBrowser", and "Structure".

**National Center for Biotechnology Information**  
National Library of Medicine      National Institutes of Health

PubMed   All Databases   **BLAST**   OMIM   Books   TaxBrowser   Structure

Search All Databases for  Go

**SITE MAP**  
Alphabetical List  
Resource Guide

**About NCBI**  
An introduction to NCBI

**GenBank**  
Sequence submission support and software

**Literature databases**  
PubMed, OMIM, Books, and PubMed Central

**Molecular databases**  
Sequences, structures, and taxonomy

**Genomic biology**  
The human genome, whole genomes, and

**Try NCBI's new home page!** NEW  
NCBI's [new home page and site guide](#) provide a catalog of NCBI resources, a consistent menu bar at the top of each page, and step-by-step instructions for common tasks. The page you are now reading will soon be replaced by the new design.

**Hot Spots**

- ▶ Clusters of orthologous groups
- ▶ Coffee Break, Genes & Disease, NCBI Handbook
- ▶ Electronic PCR
- ▶ Entrez Home
- ▶ Entrez Tools
- ▶ Gene expression omnibus (GEO)
- ▶ Human genome resources
- ▶ Influenza Virus Resource

**What does NCBI do?**

Established in 1988 as a national resource for molecular biology information, NCBI creates public databases, conducts research in computational biology, develops software tools for analyzing genome data, and disseminates biomedical information - all for the better understanding of molecular processes affecting human health and

(<http://www.ncbi.nlm.nih.gov/>)

# BLAST in practice

BLAST: Basic Local Alignment Search Tool

BLAST Basic Local Alignment Search Tool

Home Recent Results Saved Strategies Help

My NCBI Sign In Register

NCBI/ BLAST Home

BLAST finds regions of similarity between biological sequences. [more...](#)

**New** Aligning Multiple Protein Sequences? Try the **COBALT Multiple Alignment Tool**. [Go](#)

### BLAST Assembled Genomes

Choose a species genome to search, or [list all genomic BLAST databases](#).

- Human
- Mouse
- Rat
- Arabidopsis thaliana*
- Oryza sativa*
- Bos taurus*
- Danio rerio*
- Drosophila melanogaster*
- Gallus gallus*
- Pan troglodytes*
- Microbes*
- Apis mellifera*

### Basic BLAST

Choose a BLAST program to run.

- [nucleotide blast](#) Search a nucleotide database using a nucleotide query  
*Algorithms: blastn, megablast, discontinuous megablast*
- [protein blast](#) Search protein database using a protein query  
*Algorithms: blastp, psi-blast, phr...*
- [blastx](#) Search protein database using a translated nucleotide query
- [tblastn](#) Search translated nucleotide database using a protein query
- [tblastx](#) Search translated nucleotide database using a translated nucleotide query

### Specialized BLAST

**News**

[BLAST 2.2.22 now available](#)

This release includes new BLAST+ command-line applications.  
Mon, 19 Oct 2009 11:00:00 EST

[More BLAST news...](#)

**Tip of the Day**

[Use Genomic BLAST to see the genomic context](#)

If you are interested in the evolution of a particular gene or gene family it is often interesting to examine the Intron-exon structure even across species.

[More tips...](#)

## BLAST in practice

The screenshot displays the NCBI BLAST web interface. At the top, there is a navigation bar with the BLAST logo, the title 'Basic Local Alignment Search Tool', and links for 'Home', 'Recent Results', 'Saved Strategies', and 'Help'. On the right side of the navigation bar, there are links for 'My NCBI', 'Sign In', and 'Register'.

Below the navigation bar, there is a breadcrumb trail: 'NCBI/ BLAST/ blastn suite'. A sub-navigation bar contains tabs for 'blastn', 'blastp', 'blastx', 'tblastn', and 'tblastx', with 'blastn' currently selected.

The main content area is divided into two main sections:

- Enter Query Sequence:** This section contains a large text input field for 'Enter accession number, gi, or FASTA sequence'. To the right of this field are 'Clear' and 'Query subrange' links. Below the 'Query subrange' link are 'From' and 'To' input fields. Below the main input field, there is an 'Or, upload file' section with a file input field and a 'Browse...' button. Below that is a 'Job Title' input field with the placeholder text 'Enter a descriptive title for your BLAST search'. At the bottom of this section is a checkbox labeled 'Align two or more sequences'.
- Choose Search Set:** This section contains a 'Database' section with three radio buttons: 'Human genomic + transcript' (selected), 'Mouse genomic + transcript', and 'Others (nr etc.)'. Below the radio buttons is a dropdown menu currently showing 'Human genomic plus transcript (Human G+T)'. Below the dropdown menu are two checkboxes: 'Exclude Optional' (unchecked) and 'Sequences from uncultured bacteria' (unchecked). Below the checkboxes is an 'Entrez Query' input field with the placeholder text 'Enter an Entrez query to limit search'.

At the bottom of the form, there is a 'Program Selection' section.



# BLAST in practice

The screenshot shows the NCBI Nucleotide BLAST interface. At the top, the NCBI logo and 'Nucleotide' search type are visible. The search query is 'CARD15'. Below the search bar, a message states 'History has expired.' and 'Found 193 nucleotide sequences.' The results are displayed in a 'Summary' view, showing 20 items per page. The first three results are:

- 1:** [AB527390](#) Reports: Synthetic construct DNA, clone: pF1KE0396, Homo sapiens CARD15 gene for nucleotide-binding oligomerization domain containing 2, without stop codon, in Flexi system gi261858063|dbj|AB527390.1|[261858063]
- 2:** [NG\\_007508](#) Reports Links: Homo sapiens nucleotide-binding oligomerization domain containing 2 (NOD2) on chromosome 16 gi172073161|ref|NG\_007508.1|[172073161]
- 3:** [NM\\_000579](#) Reports Links: Homo sapiens chemokine (C-C motif) receptor 5 (CCR5), transcript variant A, mRNA

On the right side, there are two panels: 'Top Organisms [Tree]' showing a list of species like Homo sapiens (51), Pan troglodytes (15), and Saguinus oedipus (15); and 'Recent Activity' showing a search for 'CARD15 (188)' in the 'Nucleotide' database.

## BLAST in practice

- We will use the sequence above as a *query* sequence, and use blast to compare the query sequence to the GenBank database. The actual analysis will be run on a massively parallel supercomputer operated by NCBI as a service to the research community. There are several ways to submit searches to the blast server; we will use the web interface.
- First, copy the sequence. Then go to the NCBI web site (<http://www.ncbi.nlm.nih.gov/>), and follow the link for BLAST on the NCBI home page, and then the link for Standard nucleotide-nucleotide BLAST [blastn].
- In the space provided, paste the sequence and then click on the button that says BLAST!

## BLAST in practice

NCBI Blast

BLAST Basic Local Alignment Search Tool

Home Recent Results Saved Strategies Help

My NCBI [Sign in] [Register]

NCBI/ BLAST/ Format Request

Job Title: Nucleotide Sequence (700 letters)

Request ID	EB14S0ME01N
Status	Searching
Time since submission	00:00:00

This page will be automatically updated in 1 seconds until search is done

Copyright | Disclaimer | Privacy | Accessibility | Contact | Send Feedback

NCBI | NLM | NIH | DARS

- The initial BLAST page is replaced with a page called "formatting BLAST."
- This page provides you with a blast ID number, an estimate of how long it will take for the results to be returned, and some formatting options.
- With other sequences, the waiting can be extensive. There is no problem to explore other sites or to read the BLAST overview at

[http://www.ncbi.nlm.nih.gov/BLAST/blast\\_overview.html](http://www.ncbi.nlm.nih.gov/BLAST/blast_overview.html)



# BLAST in practice

**BLAST** Basic-Local Alignment Search Tool

Home Recent Results Saved Strategies Help

My NCBI [Sign In] [Register]

NCBI/ BLAST/ blastn suite/ Formatting Results - EB14S0ME01N

[Edit and Resubmit](#) [Save Search Strategies](#) [Formatting options](#) [Download](#)

**Nucleotide Sequence (700 letters)**

<b>Query ID</b>  cl 35529	<b>Database Name</b> dbindex/9606/allcontig_and_rna
<b>Description</b> None	<b>Description</b> Human build 37 RNA, GRCh37, and HuRef assemblies
<b>Molecule type</b> nucleic acid	<b>Program</b> BLASTN 2.2.22+ <a href="#">Citation</a>
<b>Query Length</b> 700	

Other reports: [Search Summary](#) [Taxonomy reports](#) [Distance tree of results](#) [Human genome view](#)

**Graphic Summary**

Distribution of 3 Blast Hits on the Query Sequence

Mouse over to see the define, click to show alignments

Color key for alignment scores	
Score Range	Color
<40	Black
40-50	Blue
50-80	Green
80-200	Pink
>=200	Red

Query

0 100 200 300 400 500 600 700

**Descriptions**

Legend for links to other resources: [U](#) UniGene [E](#) GEO [G](#) Gene [S](#) Structure [M](#) Map Viewer

## BLAST in practice

- The default layout of the NCBI BLAST result is a graphical representation of the hits found.
- This graphical output gives a quick overview of the query sequence (in our example 700 letters long) and the resulting hit sequences.
- The hits are color coded according to the obtained alignment scores
- Relevant questions include:
  - What inferences about this sequence can you make from this information?
  - What is the identity of the sequence?
  - What gene do you think it encodes?
  - What organism do you think it comes from?
  - How reliable do you think this inference is? Why?

# BLAST in practice

▼ Descriptions

Legend for links to other resources: [U](#) UniGene [E](#) GEO [G](#) Gene [S](#) Structure [M](#) Map Viewer

Sequences producing significant alignments:  
(Click headers to sort columns)

Accession	Description	Max score	Total score	Query coverage	E value	Max ident	Links
<b>Transcripts</b>							
<a href="#">NM_022162.1</a>	Homo sapiens nucleotide-binding oligomerization domain containing 2 (NOD2), mRNA	<a href="#">333</a>	333	25%	2e-88	100%	<a href="#">G</a> <a href="#">M</a>
<b>Genomic sequences [show first]</b>							
<a href="#">NT_010498.15</a>	Homo sapiens chromosome 16 genomic contig, GRCh37 reference primi	<a href="#">1293</a>	1293	100%	0.0	100%	
<a href="#">NW_001838288.2</a>	Homo sapiens chromosome 16 genomic contig, alternate assembly (ba	<a href="#">1293</a>	1293	100%	0.0	100%	

▼ Alignments  Select All [Get selected sequences](#) [Distance tree of results](#) **NEW**

```

> ref|NM\_022162.1 G M Homo sapiens nucleotide-binding oligomerization domain containing
2 (NOD2), mRNA
Length=4485

  GENE ID: 64127 NOD2 | nucleotide-binding oligomerization domain containing 2
[Homo sapiens] (Over 100 PubMed links)

Score = 333 bits (180), Expect = 2e-88
Identities = 180/180 (100%), Gaps = 0/180 (0%)
Strand=Plus/Plus

Query  1  GTAGACAGATCCAGGCTCACCAGTCCTGTGCCACTGGGCTTTTGGCGTTCTGCACAAGGC  60
      |||
Sbjct  1  GTAGACAGATCCAGGCTCACCAGTCCTGTGCCACTGGGCTTTTGGCGTTCTGCACAAGGC  60

Query  61  CTACCCGCAGATGCCATGCTGCTCCCCCAGCCTAATGGGCTTTGATGGGGGAAGAGGGT  120
      |||
Sbjct  61  CTACCCGCAGATGCCATGCTGCTCCCCCAGCCTAATGGGCTTTGATGGGGGAAGAGGGT  120
    
```

# BLAST in practice

NCBI Entrez Gene

Search Gene for 11545911[NUID] [Go] [Clear] [Save Search]

Display: Full Report Show: 20 Sort by: Relevance Send to: [v]

All: 1 Current Only: 1 Genes Genomes: 1 SNP GeneView: 1

1: NOD2 nucleotide-binding oligomerization domain containing 2 [ *Homo sapiens* ]  
 GeneID: 64127 updated 16-Oct-2009

**Summary**

<b>Official Symbol</b>	NOD2	provided by <a href="#">HGNC</a>
<b>Official Full Name</b>	nucleotide-binding oligomerization domain containing 2	provided by <a href="#">HGNC</a>
<b>Primary source</b>	<a href="#">HGNC:5331</a>	
<b>See related</b>	<a href="#">Ensembl:ENSG00000167207</a> ; <a href="#">HPRD:Q5810</a> ; <a href="#">MIM:605956</a>	
<b>Gene type</b>	protein coding	
<b>RefSeq status</b>	REVIEWED	
<b>Organism</b>	<a href="#">Homo sapiens</a>	
<b>Lineage</b>	<i>Eukaryota; Metazoa; Chordata; Craniata; Vertebrata; Euteleostomi; Mammalia; Eutheria; Euarchontoglires; Primates; Haplorrhini; Catarrhini; Hominidae; Homo</i>	
<b>Also known as</b>	CD; ACUG; BLAU; IBD1; NLRC2; NOD2B; CARD15; CLR16.3; PSORAS1; NOD2	
<b>Summary</b>	This gene is a member of the Nod1/Apaf-1 family and encodes a protein with two caspase recruitment (CARD) domains and six leucine-rich repeats (LRRs). The protein is primarily expressed in the peripheral blood leukocytes. It plays a role in the immune response to intracellular bacterial lipopolysaccharides (LPS) by recognizing the muramyl dipeptide (MDP) derived from them and activating the NFkB protein. Mutations in this gene have been associated with Crohn disease and Blau syndrome. [provided by RefSeq]	

**Entrez Gene Home**

**Table Of Contents**

- Summary
- Genomic regions, transcripts...
- Genomic context
- Bibliography
- Interactions
- General gene information
- General protein information
- Reference Sequences
- Related Sequences
- Additional Links

**Links** Explain

- Order cDNA clone
- Books
- CCDS
- Conserved Domains
- Genome
- GEO Profiles
- HomoloGene
- Map Viewer
- Nucleotide
- OMIM
- BioAssay, by Gene target
- PubChem Compound

# BLAST in practice

▼ Descriptions

Legend for links to other resources: [U](#) UniGene [E](#) GEO [G](#) Gene [S](#) Structure [M](#) Map Viewer

Sequences producing significant alignments:  
(Click headers to sort columns)

Accession	Description	Max score	Total score	Query coverage	E value	Max ident	Links
<b>Transcripts</b>							
<a href="#">NM_022162.1</a>	Homo sapiens nucleotide-binding oligomerization domain containing 2 (NOD2), mRNA	<a href="#">333</a>	333	25%	2e-88	100%	<a href="#">G</a> <a href="#">M</a>
<b>Genomic sequences [show first]</b>							
<a href="#">NT_010498.15</a>	Homo sapiens chromosome 16 genomic contig, GRCh37 reference primary assembly (bam)	<a href="#">1293</a>	1293	100%	0.0	100%	
<a href="#">NW_001838288.2</a>	Homo sapiens chromosome 16 genomic contig, alternate assembly (bam)	<a href="#">1293</a>	1293	100%	0.0	100%	

▼ Alignments  Select All [Get selected sequences](#) [Distance tree of results](#) **NEW**

```

> ref|NM\_022162.1 G M Homo sapiens nucleotide-binding oligomerization domain containing
2 (NOD2), mRNA
Length=4485

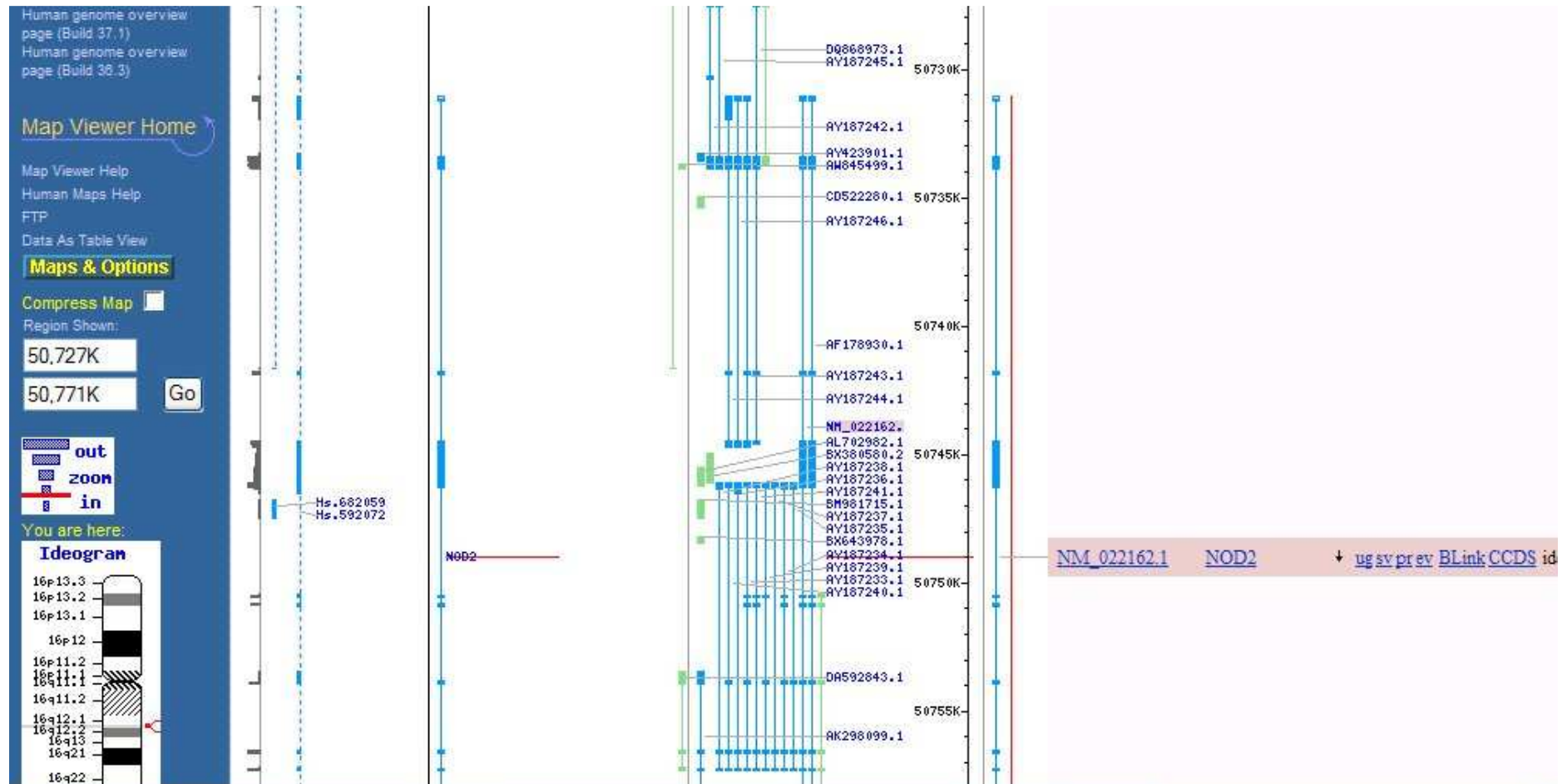
  GENE ID: 64127 NOD2 | nucleotide-binding oligomerization domain containing 2
[Homo sapiens] (Over 100 PubMed links)

Score = 333 bits (180), Expect = 2e-88
Identities = 180/180 (100%), Gaps = 0/180 (0%)
Strand=Plus/Plus

Query  1  GTAGACAGATCCAGGCTCACCAGTCCTGTGCCACTGGGCTTTTGGCGTTCTGCACAAGGC  60
      |||
Sbjct  1  GTAGACAGATCCAGGCTCACCAGTCCTGTGCCACTGGGCTTTTGGCGTTCTGCACAAGGC  60

Query  61  CTACCCGCAGATGCCATGCTGCTCCCCCAGCCTAATGGGCTTTGATGGGGGAAGAGGGT  120
      |||
Sbjct  61  CTACCCGCAGATGCCATGCTGCTCCCCCAGCCTAATGGGCTTTGATGGGGGAAGAGGGT  120
    
```

# BLAST in practice



# BLAST in practice

▼ Descriptions

Legend for links to other resources: [U](#) UniGene [E](#) GEO [G](#) Gene [S](#) Structure [M](#) Map Viewer

Sequences producing significant alignments:  
(Click headers to sort columns)

Accession	Description	Max score	Total score	Query coverage	E value	Max ident	Links
<b>Transcripts</b>							
<a href="#">NM_022162.1</a>	Homo sapiens nucleotide-binding oligomerization domain containing 2 (NOD2), mRNA	<a href="#">333</a>	333	25%	2e-88	100%	<a href="#">G</a> <a href="#">M</a>
<b>Genomic sequences [show first]</b>							
<a href="#">NT_010498.15</a>	Homo sapiens chromosome 16 genomic contig, GRCh37 reference primary assembly	<a href="#">1293</a>	1293	100%	0.0	100%	
<a href="#">NW_001838288.2</a>	Homo sapiens chromosome 16 genomic contig, alternate assembly (BAC)	<a href="#">1293</a>	1293	100%	0.0	100%	

▼ Alignments  Select All [Get selected sequences](#) [Distance tree of results](#) **NEW**

```

> ref|NM\_022162.1| GM Homo sapiens nucleotide-binding oligomerization domain containing
2 (NOD2), mRNA
Length=4485

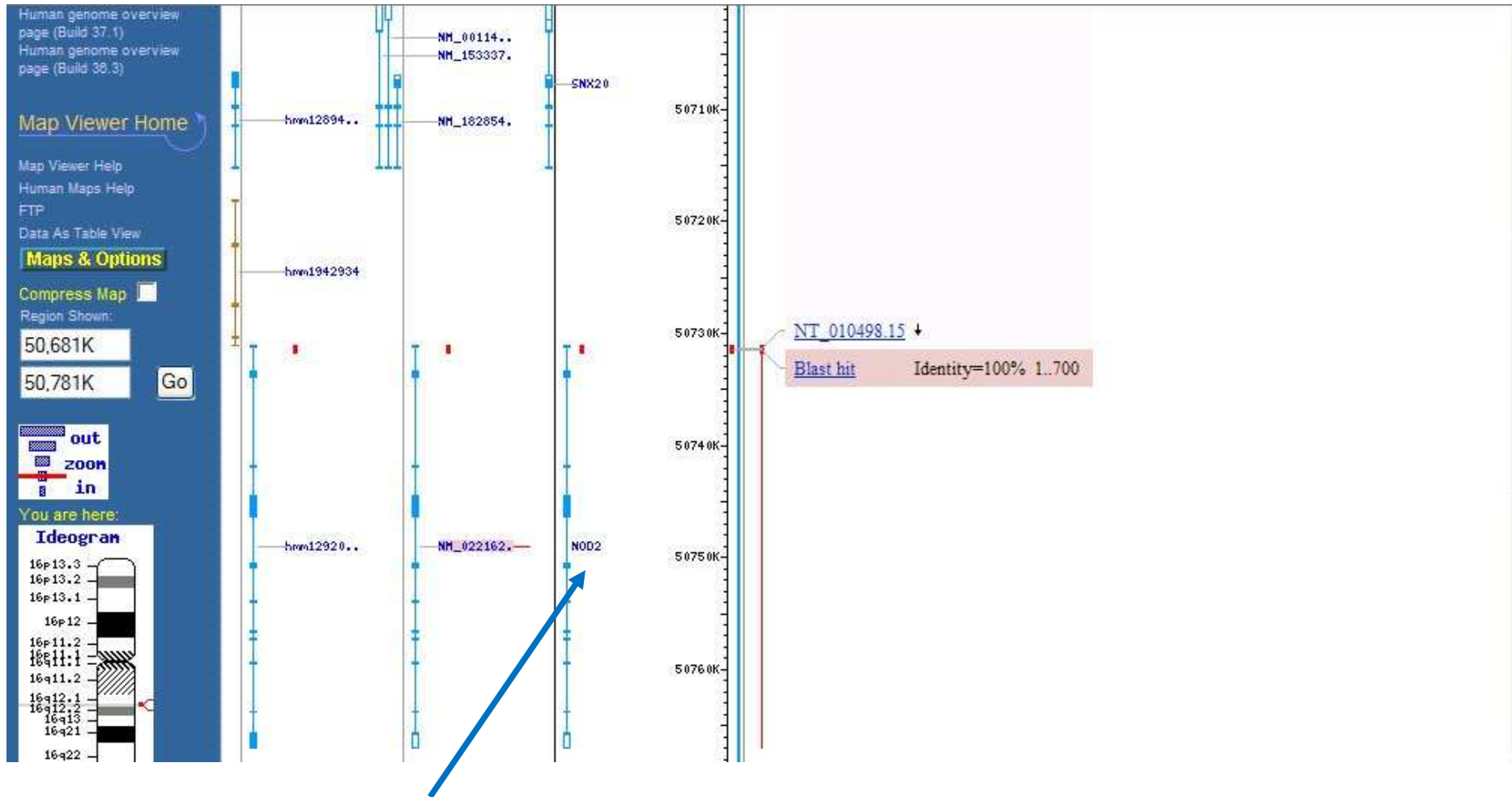
  GENE ID: 64127 NOD2 | nucleotide-binding oligomerization domain containing 2
[Homo sapiens] (Over 100 PubMed links)

Score = 333 bits (180), Expect = 2e-88
Identities = 180/180 (100%), Gaps = 0/180 (0%)
Strand=Plus/Plus

Query  1  GTAGACAGATCCAGGCTCACCAGTCCTGTGCCACTGGGCTTTTGGCGTTCTGCACAAGGC  60
      |||
Sbjct  1  GTAGACAGATCCAGGCTCACCAGTCCTGTGCCACTGGGCTTTTGGCGTTCTGCACAAGGC  60

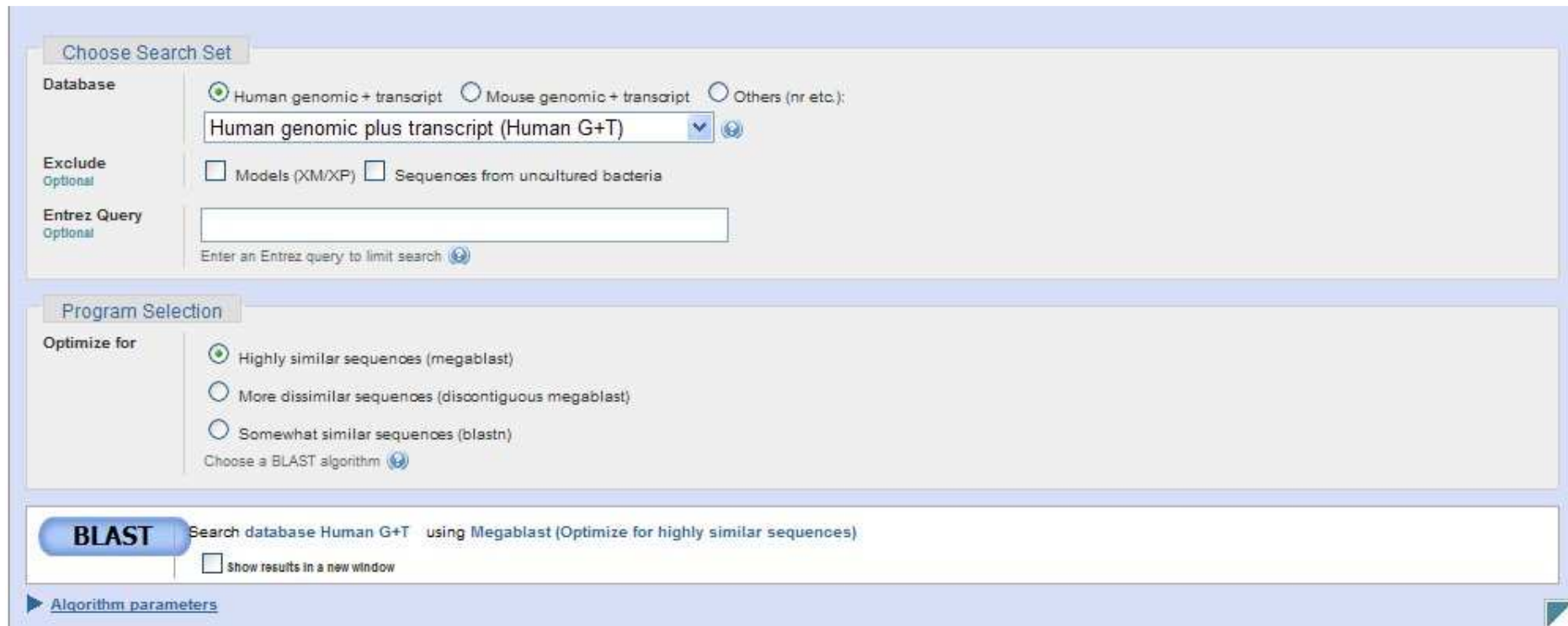
Query  61  CTACCCGCAGATGCCATGCTGCTCCCCCAGCCTAATGGGCTTTGATGGGGGAAGAGGGT  120
      |||
Sbjct  61  CTACCCGCAGATGCCATGCTGCTCCCCCAGCCTAATGGGCTTTGATGGGGGAAGAGGGT  120
    
```

# BLAST in practice





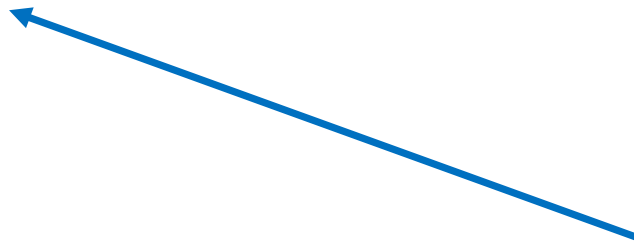
## BLAST in practice



The screenshot displays the NCBI BLAST search interface. It is divided into three main sections:

- Choose Search Set:** This section allows users to select a database and optional filters. The "Database" section has three radio buttons: "Human genomic + transcript" (selected), "Mouse genomic + transcript", and "Others (nr etc.);". Below this is a dropdown menu showing "Human genomic plus transcript (Human G+T)". The "Exclude" section has two checkboxes: "Models (XM/XP)" and "Sequences from uncultured bacteria", both of which are unchecked. The "Entrez Query" section has a text input field and a label "Enter an Entrez query to limit search".
- Program Selection:** This section allows users to optimize the search. It has three radio buttons: "Highly similar sequences (megablast)" (selected), "More dissimilar sequences (discontiguous megablast)", and "Somewhat similar sequences (blastn)". Below these is a link "Choose a BLAST algorithm".
- BLAST:** This section shows the search parameters: "Search database Human G+T using Megablast (Optimize for highly similar sequences)". There is a checkbox "show results in a new window" which is unchecked.

At the bottom of the interface, there is a link "Algorithm parameters" with a right-pointing triangle icon.



## BLAST in practice

Algorithm parameters Note: Parameter values that differ from the default are highlighted in yellow

**General Parameters**

Max target sequences: 100 Select the maximum number of aligned sequences to display

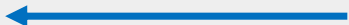
Short queries:  Automatically adjust parameters for short input sequences

Expect threshold: 10

Word size: 28

**Scoring Parameters**

Match/Mismatch Scores: 1,-2

Gap Costs: Linear 

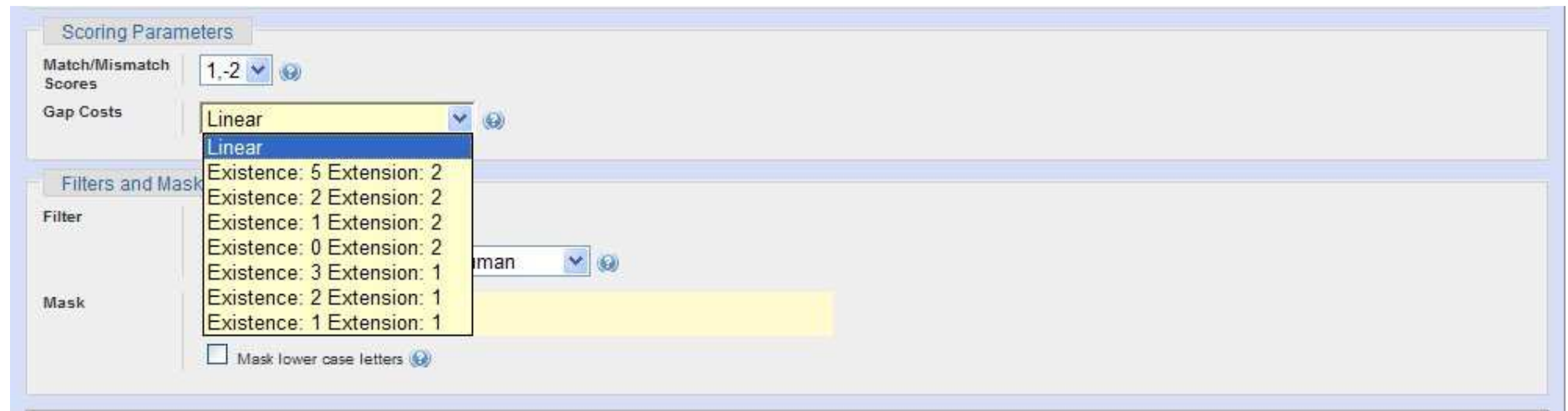
**Filters and Masking**

Filter:  Low complexity regions  
 Species-specific repeats for: Human

Mask:  Mask for lookup table only  
 Mask lower case letters

**BLAST** Search database Human G+T using Megablast (Optimize for highly similar sequences)

## BLAST in practice



- Can you make the link with earlier derivations in this chapter?

$$\omega(k) = -\alpha - \beta(k - 1)$$

## BLAST in practice

- Consider the same genomic sequence as before

BLAST: Basic Local Alignment Search Tool

BLAST Basic Local Alignment Search Tool

Home Recent Results Saved Strategies Help

My NCBI [Sign In] [Register]

NCBI/ BLAST Home

BLAST finds regions of similarity between biological sequences. [more...](#)

**New** Aligning Multiple Protein Sequences? Try the **COBALT Multiple Alignment Tool**. [Go](#)

### BLAST Assembled Genomes

Choose a species genome to search, or [list all genomic BLAST databases](#).

- Human
- Mouse
- Rat
- Arabidopsis thaliana*
- Oryza sativa*
- Bos taurus*
- Danio rerio*
- Drosophila melanogaster*
- Gallus gallus*
- Pan troglodytes*
- Microbes*
- Apis mellifera*

### Basic BLAST

Choose a BLAST program to run.

- [nucleotide blast](#) Search a nucleotide database using a nucleotide query  
*Algorithms: blastn, megablast, discontinuous megablast*
- [protein blast](#) Search protein database using a protein query  
*Algorithms: blastp, psi-blast, phi-blast*
- [blastx](#) Search protein database using a translated nucleotide query
- [tblastn](#) Search translated nucleotide database using a protein query
- [tblastx](#) Search translated nucleotide database using a translated nucleotide query

### Specialized BLAST

#### News

[BLAST 2.2.22 now available](#)

This release includes new BLAST+ command-line applications.  
Mon, 19 Oct 2009 11:00:00 EST

[More BLAST news...](#)

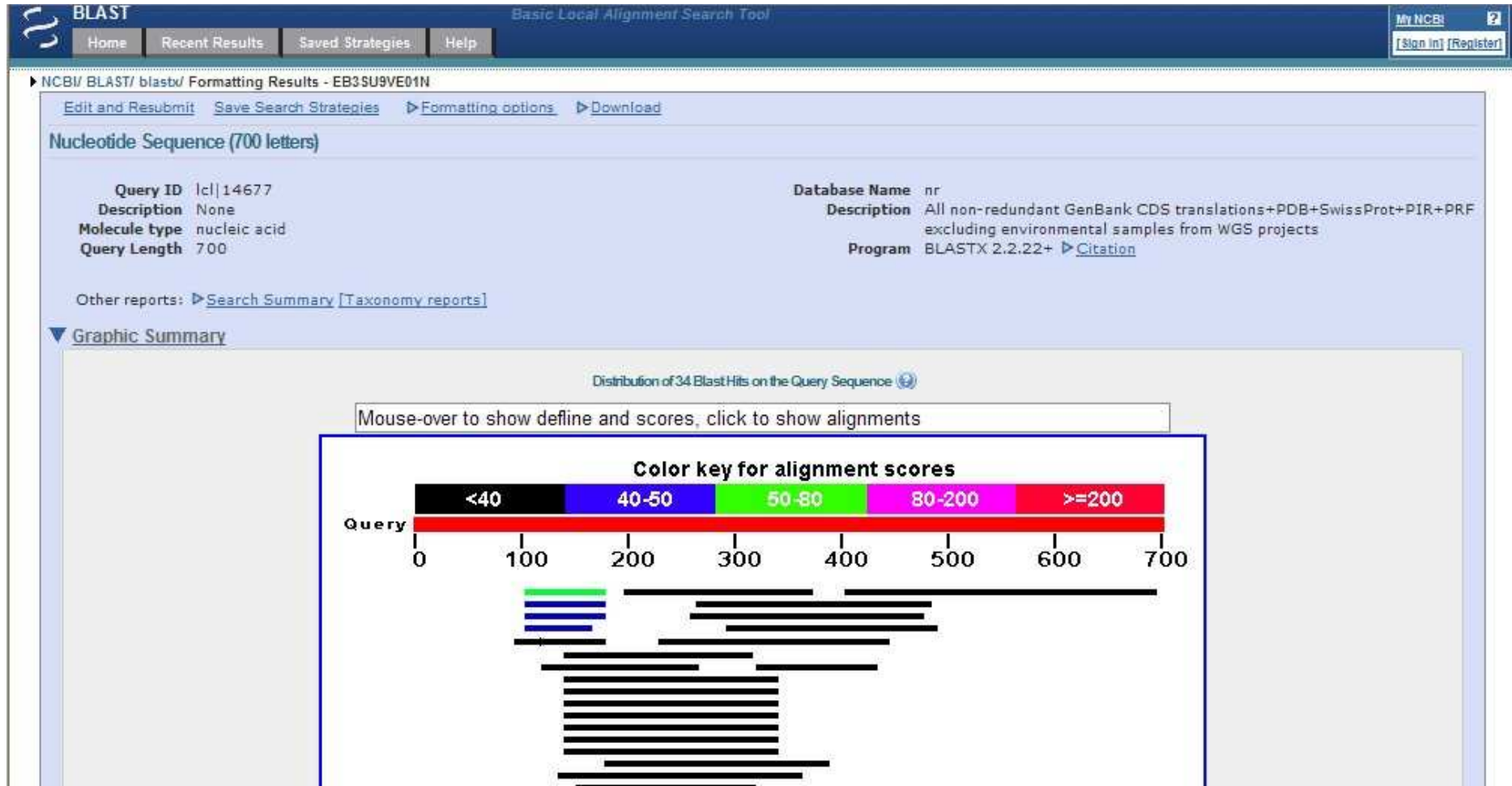
#### Tip of the Day

[Use Genomic BLAST to see the genomic context](#)

If you are interested in the evolution of a particular gene or gene family it is often interesting to examine the intro-exon structure even across species.

[More tips...](#)

# BLAST in practice



# BLAST in practice

Entrez Gene: NOD2 nucleotide-binding oligomerization...

---

**NCBI Reference Sequences (RefSeq)**

RefSeqs maintained independently of Annotated Genomes

These reference sequences exist independently of genome builds. [Explain](#)

**Genomic**

1. **NG\_007508.1 RefSeqGene**

Range: 5001..40938  
 Download: [GenBank](#), [FASTA](#), [Sequence Viewer \(Graphics\)](#)

**mRNA and Protein(s)**

1. **NM\_022162.1 .NP\_071445.1 nucleotide-binding oligomerization domain containing 2**

Source sequence(s): [AF178930](#)  
 Consensus CDS: [CCDS10746.1](#)  
 UniProtKB/Swiss-Prot: [Q9HC19](#)  
 Conserved Domains (4) [summary](#)

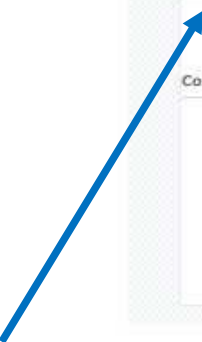
<a href="#">c02423</a> Location: 767-1032 Blast Score: 247	LRR_RI; Leucine-rich repeats (LRRs), ribonuclease inhibitor (RI)-like subfamily. LRRs are 20-29 residue sequence motifs present in many proteins that participate in protein-protein interactions and have different functions and cellular locations. LRRs...
<a href="#">c02470</a> Location: 31-104 Blast Score: 151 Location: 132-204 Blast Score: 116	CARD; Caspase recruitment domain
<a href="#">c03089</a> Location: 293-463 Blast Score: 332	P-loop NTPase; P-loop containing Nucleoside Triphosphate Hydrolases

**RefSeqs of Annotated Genomes: Build 37.1**

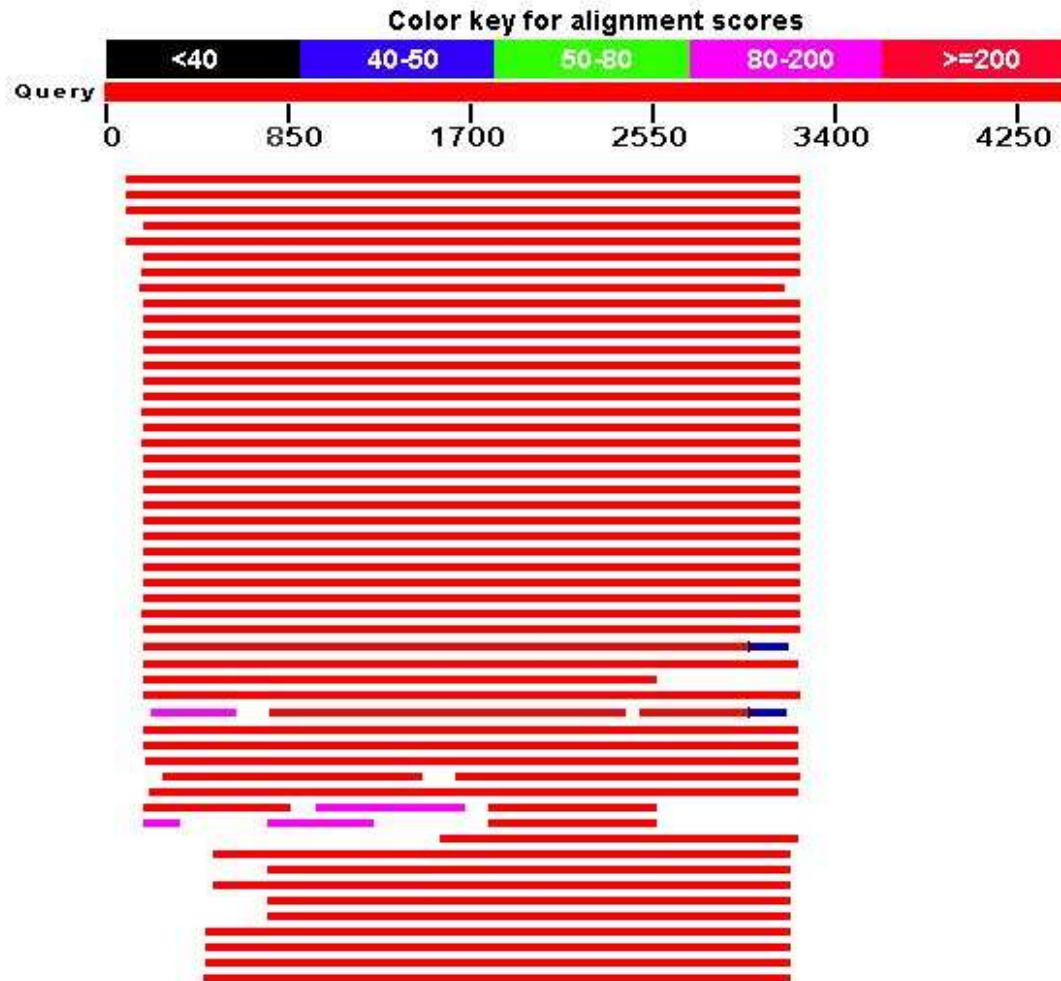
The following sections contain reference sequences that belong to a specific genome build. [Explain](#)

Genome Reference Consortium Human Build 37 (GRCh37), Primary\_Assembly

**Genomic**



## BLAST in practice



- Do a BLAST search using retrieved mRNA/protein sequence.
- Use accession number / FASTA format as input

## BLAST properties

- BLAST is extremely fast. It can be on the order of 50-100 times faster than the Smith-Waterman approach
  - Note that because of the different strategies followed by FASTA and BLAST, FASTA may be better for less similar sequences
  - For highly divergent sequences, even FASTA may perform poorly. Hence, evolutionary diverse members of the same “family” (e.g., a family of proteins; since alignments may also be performed between amino-acid sequences) may be overlooked.
  - Its main idea is built on the conjecture that homologous sequences are likely to contain a short high-scoring similarity region, a hit. Each hit gives a seed that BLAST tries to extend on both sides.
- It is preferred over FASTA for large database searches
- There exists a statistical theory for assessing significance



## BLAST properties

- Many variants exist to the initial BLAST theme ... Depending on the nature of the sequence it is possible to use different BLAST programs for the database search.
- There are five versions of the BLAST program: BLASTN, BLASTP, BLASTX, TBLASTN, TBLASTX:

Option	Query Type	DB Type	Comparison	Note
blastn	Nucleotide	Nucleotide	Nucleotide-Nucleotide	
blastp	Protein	Protein	Protein-Protein	
tblastn	Protein	Nucleotide	Protein-Protein	The database is translated into protein
blastx	Nucleotide	Protein	Protein-Protein	The queries are translated into protein
tblastx	Nucleotide	Nucleotide	Protein-Protein	The queries and database are translated into protein

## Should I use Smith-Waterman or other algorithms for sequence similarity searching?

- The Smith-Waterman algorithm is quite time demanding because of the search for optimal local alignments, and it also imposes some requirements on the computer's memory resources as the comparison takes place on a character-to-character basis.
- The fact that similarity searches using the Smith-Waterman algorithm take a lot of time often prevents this from being the first choice, even though it is the most precise algorithm for identifying homologous regions between sequences.
- A combination of the Smith-Waterman algorithm with a reduction of the search space (such as k-word identification in FASTA) may speed up the process.

## Should I use Smith-Waterman or other algorithms for sequence similarity searching?

- BLAST and FASTA are heuristic approximations of dynamic programming algorithms. These approximations are less sensitive (then f.i. Smith-Waterman) and do not guarantee to find the best alignment between two sequences. However, these methods are not as time-consuming as they reduce computation time and CPU usage [Shpaer et al., 1996].
- Hence, the researcher needs to make a choice between
  - Having a fast and effective data analysis (BLAST)
  - Reducing the risk of missing important information by using the most sensitive algorithms for data base searching (Smith-Waterman / FASTA)

## Should I use Smith-Waterman or other algorithms for sequence similarity searching?

- Through the Japanese Institute of Bioinformatics Research and Development (BIRD) a public available software version of Smith-Waterman, SSEARCH, is accessible: <http://www-btls.jst.go.jp/cgi-bin/Tools/SSEARCH/index.cgi>. There are also commercial software packages available which perform Smith-Waterman searches.
- Remember that the result of a Smith-Waterman algorithm searching will be only returning one result for each pair of compared sequences: the optimal alignment

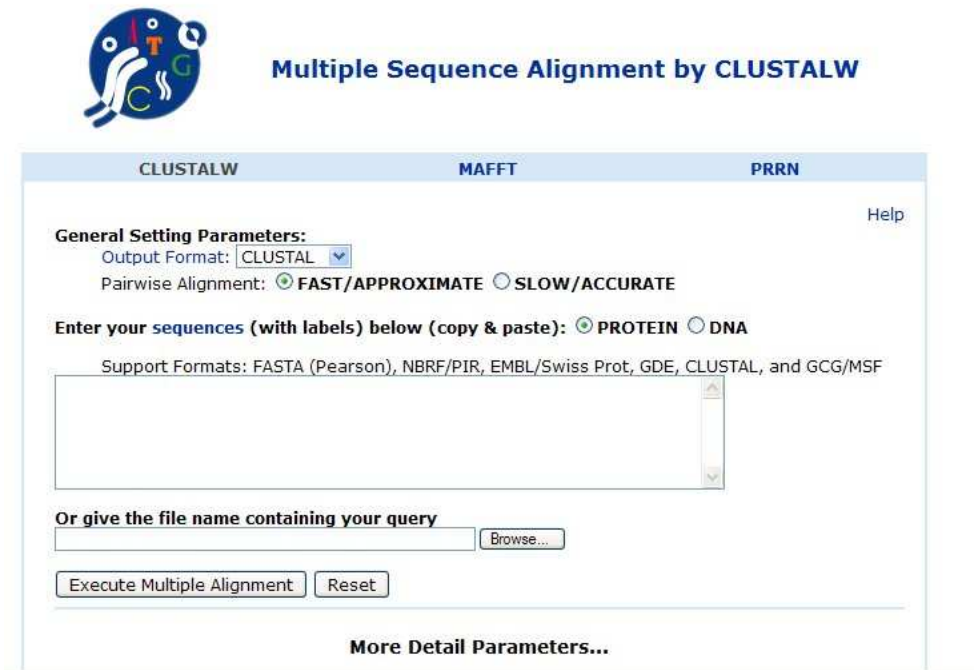
## 7. Multiple alignment

### Introduction

- In practice, real-world multiple alignment problems are usually solved with heuristics as well
- Progressive multiple alignment:
  - Choose two sequences and align them
  - Choose third sequences wrt two previous sequences and align the third against them
  - Repeat until all sequences have been aligned
  - Different options how to choose sequences and score alignments ...

## Multiple alignment in practice

- CLUSTALW, Thompson et al. NAR 1994.
  - Computes all pairwise global alignments
  - Estimates a tree (or cluster) of relationships using alignment scores
  - Collects the sequences into a multiple alignment using the tree and pairwise alignments as a guide for adding each successive sequence



The screenshot shows the web interface for multiple sequence alignment using CLUSTALW. At the top, there is a logo with a hand holding a test tube and a flask, and the text "Multiple Sequence Alignment by CLUSTALW". Below the logo, there are three tabs: "CLUSTALW", "MAFFT", and "PRRN", with "CLUSTALW" selected. The interface includes a "General Setting Parameters:" section with a "Help" link. The "Output Format:" is set to "CLUSTAL". The "Pairwise Alignment:" options are "FAST/APPROXIMATE" (selected) and "SLOW/ACCURATE". The "Enter your sequences (with labels) below (copy & paste):" section has radio buttons for "PROTEIN" (selected) and "DNA". Below this is a text area for pasting sequences, with "Support Formats: FASTA (Pearson), NBRF/PIR, EMBL/Swiss Prot, GDE, CLUSTAL, and GCG/MSF" listed. There is a "Browse..." button for file uploads. At the bottom, there are "Execute Multiple Alignment" and "Reset" buttons, and a link for "More Detail Parameters...".

(<http://align.genome.jp/>)

## Multiple alignment in practice

- T-Coffee (Tree-based Consistency Objective Function for alignment Evaluation), C. Notredame et al. JMB 2000

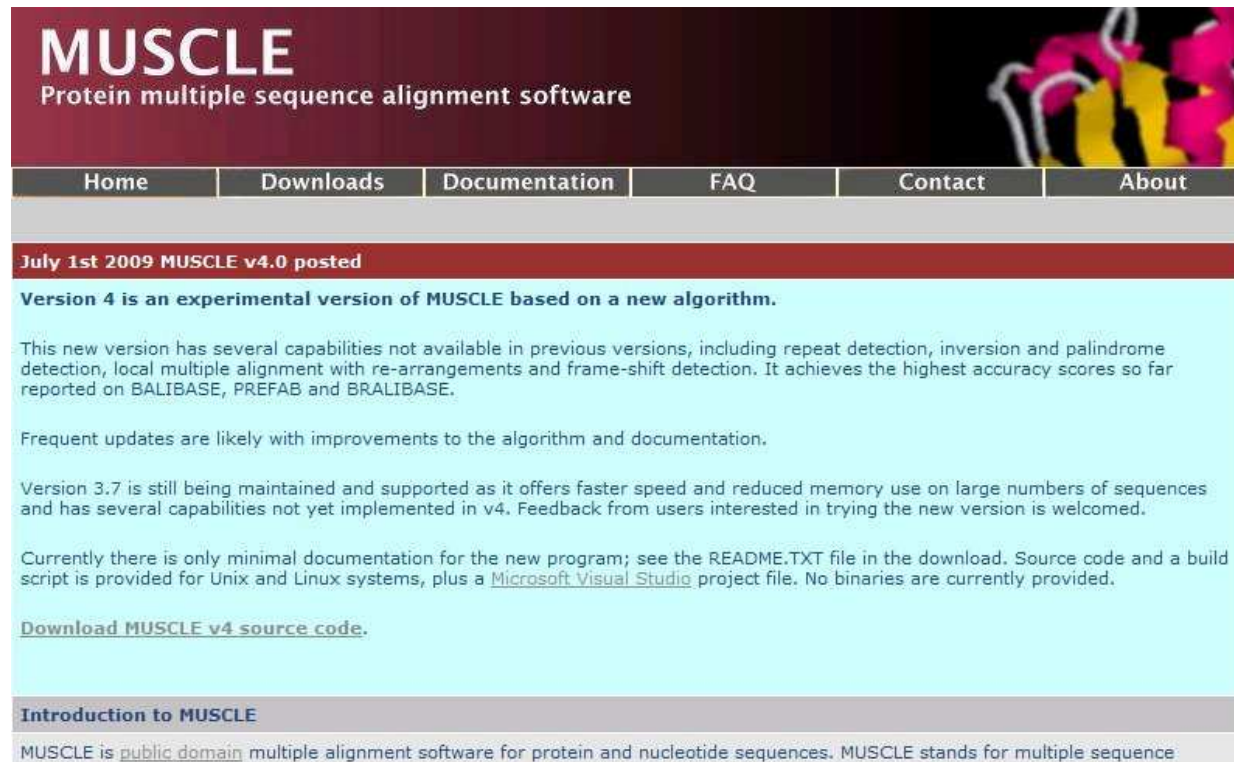


The image shows a screenshot of the T-Coffee website homepage. The page has a light orange background. At the top, the title "T-Coffee" is displayed in a large, bold, dark orange font, followed by the subtitle "Multiple Sequence Alignment Tools" in a slightly smaller, bold, dark orange font. Below the title is a horizontal navigation bar with six links: "HOME", "SERVERS", "DOWNLOAD", "DOCUMENTATION", "FAQ", and "CONTACT", all in a small, dark orange font. Below the navigation bar, there are two main content areas. The left area is titled "What is T-Coffee?" and contains two paragraphs of text. The first paragraph describes T-Coffee as a multiple sequence alignment package that can combine the output of various alignment methods (Clustal, Mafft, Probcons, Muscle...) into one unique alignment (M-Coffee). The second paragraph states that T-Coffee can align Protein, DNA, and RNA sequences and is also able to combine sequence information with protein structural information (3D-Coffee/Expresso), profile information (PSI-Coffee), or RNA secondary structures (R-Coffee). The right area is titled "SERVERS" and contains a bulleted list of links to various servers: "www.tcoffee.org", "SIB", "CNRS", "EBI", "Max Plank", "Cornell", and "CRS4".

([http://www.tcoffee.org/Projects\\_home\\_page/t\\_coffee\\_home\\_page.html](http://www.tcoffee.org/Projects_home_page/t_coffee_home_page.html))

## Multiple alignment in practice

- MUSCLE (Multiple sequence comparison by log expectation), R. Edgar NAR 2004.



**MUSCLE**  
Protein multiple sequence alignment software

Home Downloads Documentation FAQ Contact About

**July 1st 2009 MUSCLE v4.0 posted**

**Version 4 is an experimental version of MUSCLE based on a new algorithm.**

This new version has several capabilities not available in previous versions, including repeat detection, inversion and palindrome detection, local multiple alignment with re-arrangements and frame-shift detection. It achieves the highest accuracy scores so far reported on BALIBASE, PREFAB and BRALIBASE.

Frequent updates are likely with improvements to the algorithm and documentation.

Version 3.7 is still being maintained and supported as it offers faster speed and reduced memory use on large numbers of sequences and has several capabilities not yet implemented in v4. Feedback from users interested in trying the new version is welcomed.

Currently there is only minimal documentation for the new program; see the README.TXT file in the download. Source code and a build script is provided for Unix and Linux systems, plus a [Microsoft Visual Studio](#) project file. No binaries are currently provided.

[Download MUSCLE v4 source code.](#)

**Introduction to MUSCLE**

MUSCLE is [public domain](#) multiple alignment software for protein and nucleotide sequences. MUSCLE stands for multiple sequence

(<http://www.drive5.com/muscle/>)



## 8 Proof of concept

### Tests of alignment methods

- At this point, we should remind ourselves why we are performing alignments in the first place.
  - In many cases, the purpose is to identify homologs of the query sequence so that we can attribute to the query annotations associated with its homologs in the database.
- The question is, "What are the chances of finding in a database search HSPs that are not homologs?"
  - Over evolutionary time, it is possible for sequences of homologous proteins to diverge significantly. This means that to test alignment programs, some approach other than alignment scores is needed to find homologs. Often the three dimensional structures of homologs and their domain structures will be conserved ... Hence, structure can be used as a criterion for identifying homologs in a test set.

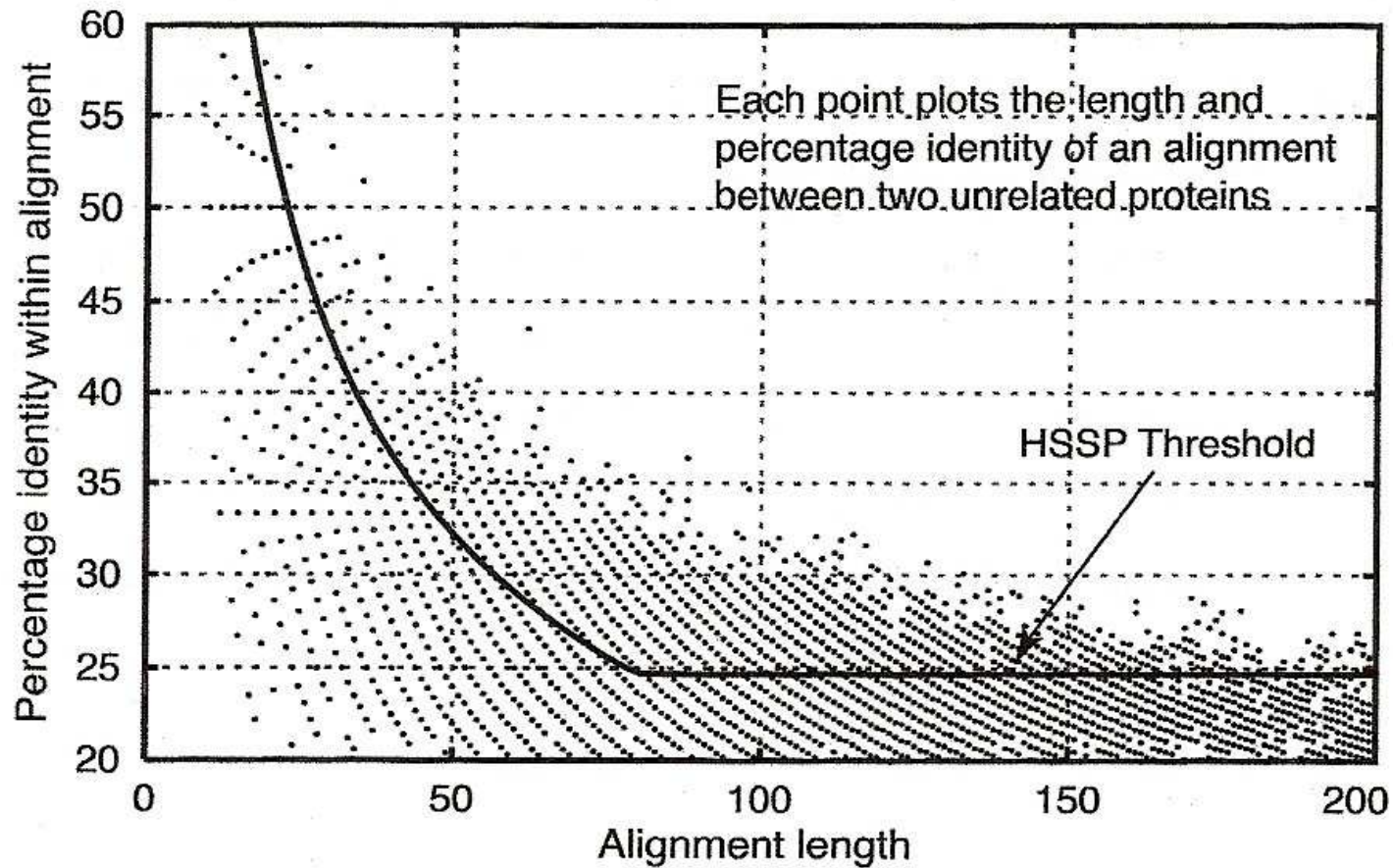
## Tests of alignment methods

- A "good" alignment program meets at least two criteria:
  - it maximizes the number of homologs found (true positives), and
  - it minimizes the number of nonhomologous proteins found (false positives).
    - Another way to describe these criteria is in terms of sensitivity and specificity (see before).
    - In this context, sensitivity is a measure of the fraction of actual homologs that are identified by the alignment program, and the specificity is a measure of the fraction of HSPs that are not actually homologs.
- Brenner et al. (1998) tested a number of different alignment approaches, including Smith- Waterman, FASTA, and an early version of BLAST. They discovered that, at best, only about 35% of homologs were detectable at a false positive error frequency of 0.1% per query sequence.

## Tests of alignment methods

- An intuitive measure of homology employed in the past was the percentage of sequence identity.
  - The rule of thumb was that sequence identities of 25%-30% in an alignment signified true homology.
  - Brenner et al. employed a database of known proteins annotated with respect to homology / nonhomology relationships to test the relationship between sequence identity and homology. Their results are shown in the figures on the next slide

## Tests of alignment methods



## Tests of alignment methods

- The figure on the previous slide shows percentage identity plotted against alignment length for proteins that are not homologs.
  - For comparison, a threshold percentage identity taken to imply similar structure is plotted as a line (see Brenner et al., 1998 for details).
  - The point is that for alignments 100 residues in length, about half of the nonhomologous proteins show more than 25% sequence identity.
  - At  $50 \pm 10$  residues of alignment length, there are a few nonhomologous proteins having over 40% sequence identity.
- This plot serves as a reminder of why methods providing detailed statistical analysis of HSPs are required, as we indicated throughout this chapter (e.g., E values and newer versions of BLAST).

## References:

- Deonier et al. *Computational Genome Analysis*, 2005, Springer.  
(Chapters 6,7)

## Background reading:

- Delsuc et al 2005. Phylogenomics and the reconstruction of the tree of life. *Nature Reviews Genetics* 6: 361-.

## In-class discussion document

- Eddy 2004. What is dynamic programming? Nature Biotechnology 22(7): 909-910.

Questions: In class reading\_5.pdf

### Preparatory Reading:

- Shriver et al 2004. Genetic ancestry and the search for personalized genetic histories. Nature Reviews Genetics 5: 611-.
- Foster et al 2004. Beyond race : towards a whole genome perspective on human populations and genetic variation. Nature Reviews Genetics 5: 790-.
- Balding 2006. A tutorial on statistical methods for population association studies. Nature Reviews Genetics 7: 781-.