Genetics and Bioinformatics GBIO0002

Archana Bhardwaj

FASTA format

- **The FASTA format is a simple and widely used format for storing biological (DNA or protein) sequences.**
- **It was first used by the FASTA program for sequence alignment.**
- **It begins with a single-line description starting with a “>” character, followed by lines of sequences.**
- **Here is an example of a FASTA file:**

```
> A06852 183 residues
MPRLFSYLLGVWLLLSQLPREIPGQSTNDFIKACGRELVRLWVEICGSVSWGRTALSLEE
PQLETGPPAETMPSSITKDAEILKMMLEFVPNLPQELKATLSERQPSLRELQQSASKDSN
LNFEEFKKIILNRQNEAEDKSLLELKNLGLDKHSRKKRLFRMTLSEKCCQVGCIRKDIARLC
```

Sequence Database

- The National Centre for Biotechnology Information (NCBI) (www.ncbi.nlm.nih.gov) in the US maintains a huge database of all the DNA and protein sequence data that has been collected, the NCBI Sequence Database

- A similar database in Europe, the European Molecular Biology Laboratory (EMBL) Sequence Database (www.ebi.ac.uk/embl)

- A similar database in Japan, the DNA Data Bank of Japan (DDBJ; www.ddbj.nig.ac.jp).

- These three databases exchange data every night, so at any one point in time, they contain almost identical data.

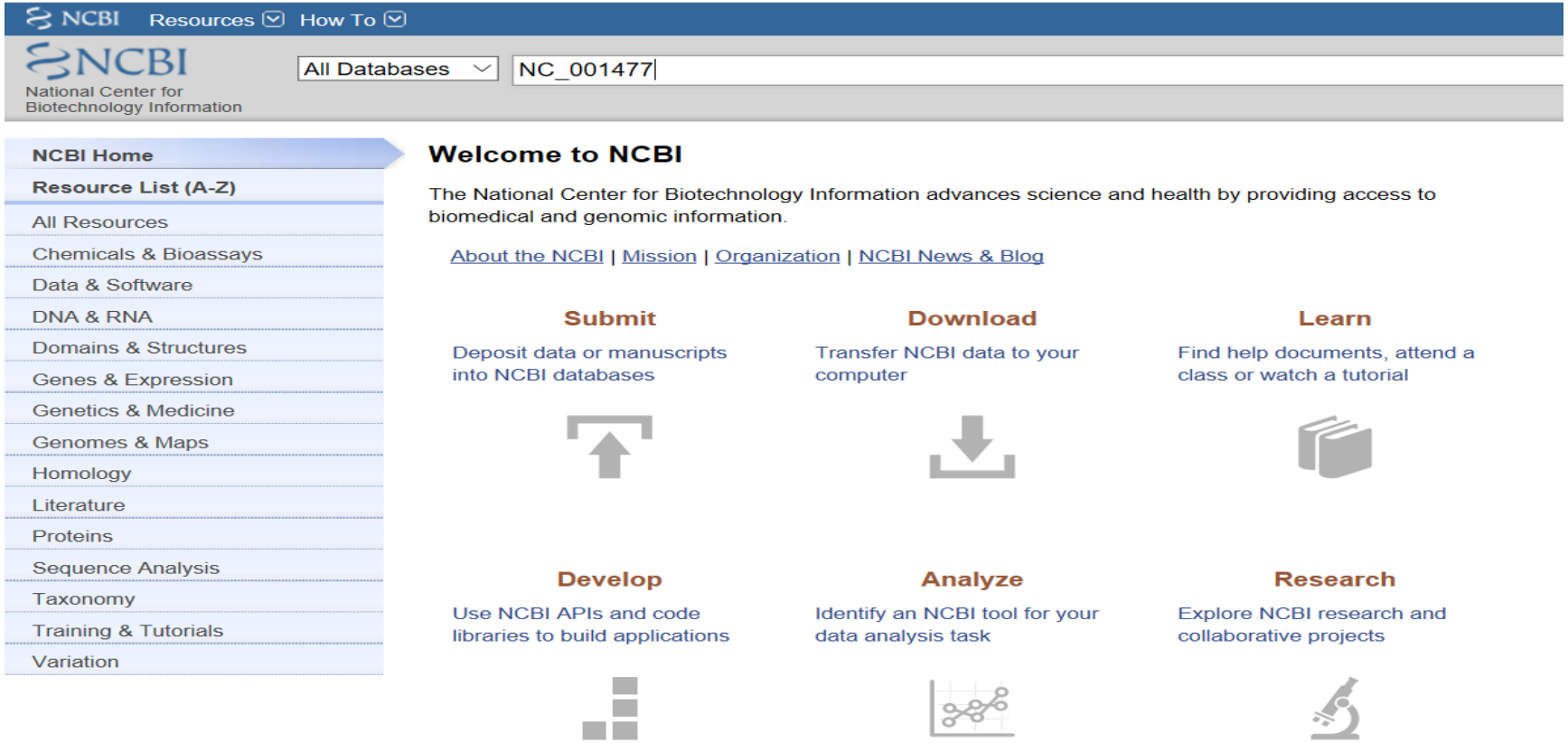
- **Each sequence in the NCBI Sequence Database is stored in a separate *record*, and is assigned a unique identifier that can be used to refer to that sequence record.**

- **The identifier is known as an *accession*, and consists of a mixture of numbers and letters.**

- **The NCBI accessions for the DNA sequences of the DEN-1, DEN-2, DEN-3, and DEN-4 Dengue viruses are NC_001477, NC_001474, NC_001475 and NC_002640, respectively.**

Retrieving genome sequence data via the NCBI website

- You can easily retrieve DNA or protein sequence data from the NCBI Sequence Database via its website www.ncbi.nlm.nih.gov



Dengue virus 1, complete genome

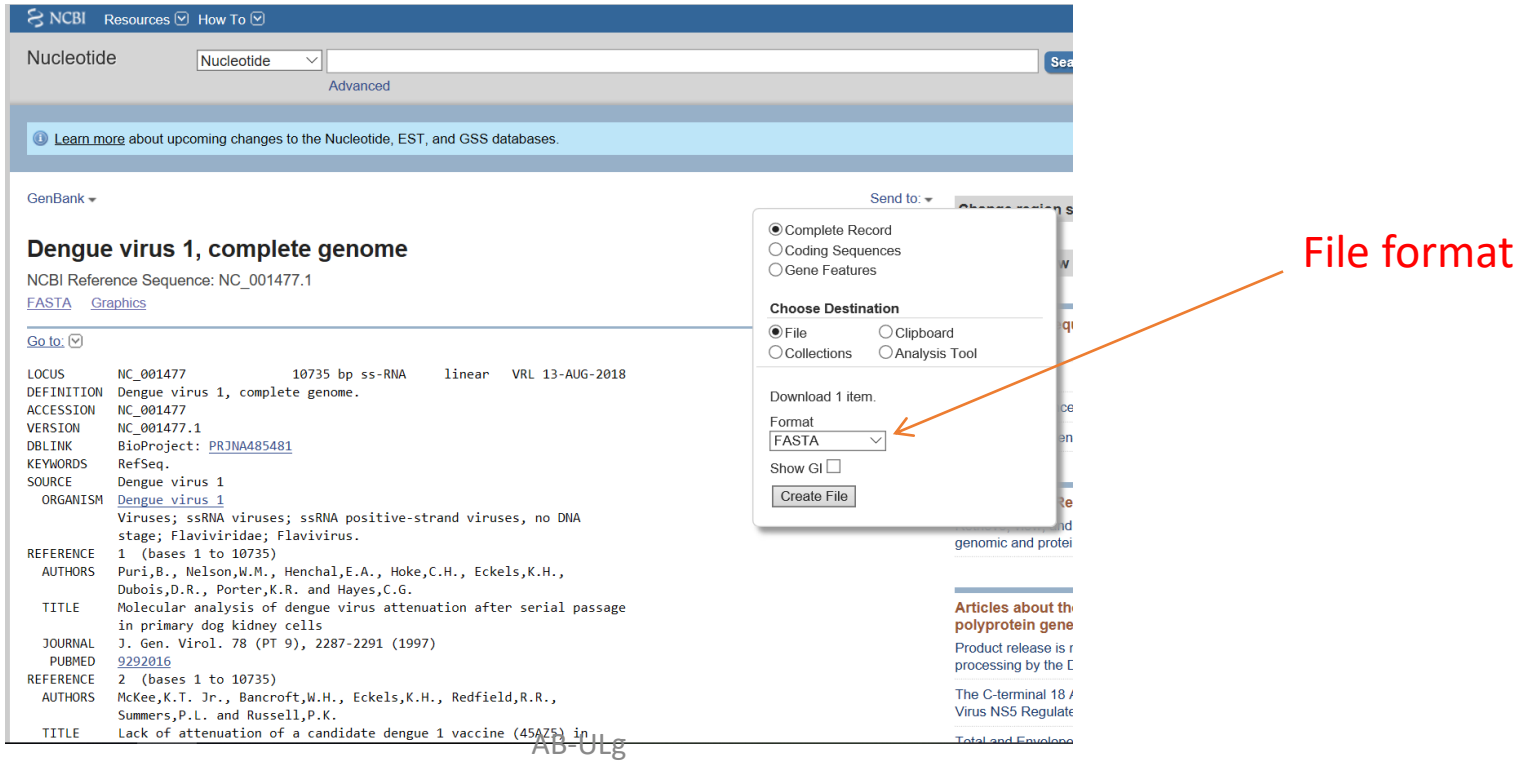
NCBI Reference Sequence: NC_001477.1

FASTA Graphics

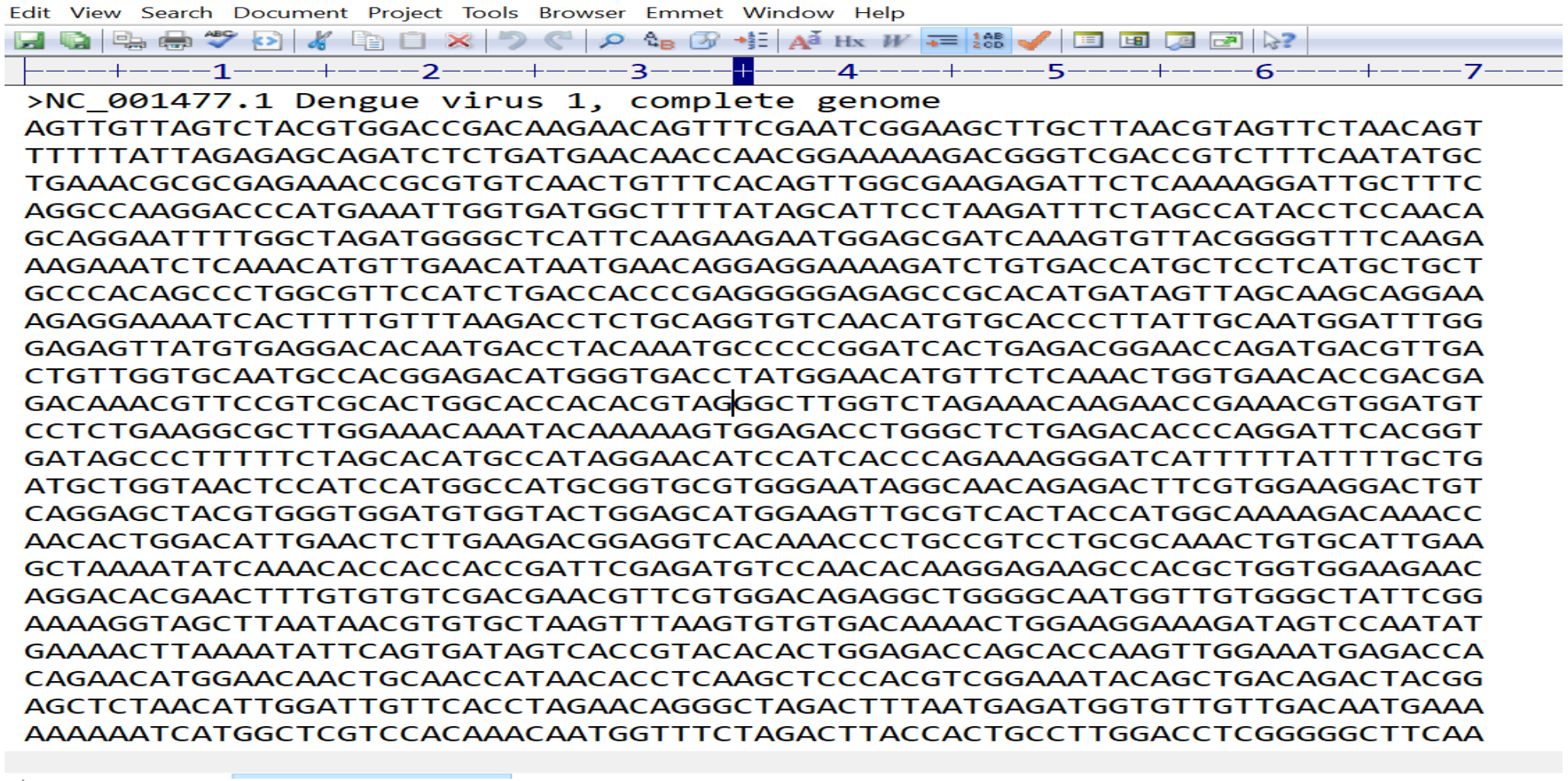
Go to:

```
LOCUS       NC_001477               10735 bp ss-RNA     linear   VRL 13-AUG-2018
DEFINITION  Dengue virus 1, complete genome.
ACCESSION   NC_001477
VERSION     NC_001477.1
DBLINK      BioProject: PRJNA485481
KEYWORDS    RefSeq.
SOURCE      Dengue virus 1
  ORGANISM  Dengue virus 1
            Viruses; ssRNA viruses; ssRNA positive-strand viruses, no DNA
            stage; Flaviviridae; Flavivirus.
REFERENCE   1  (bases 1 to 10735)
  AUTHORS   Puri,B., Nelson,W.M., Henchal,E.A., Hoke,C.H., Eckels,K.H.,
            Dubois,D.R., Porter,K.R. and Hayes,C.G.
  TITLE     Molecular analysis of dengue virus attenuation after serial passage
            in primary dog kidney cells
  JOURNAL   J. Gen. Virol. 78 (PT 9), 2287-2291 (1997)
   PUBMED   9292016
REFERENCE   2  (bases 1 to 10735)
  AUTHORS   McKee,K.T. Jr., Bancroft,W.H., Eckels,K.H., Redfield,R.R.,
            Summers,P.L. and Russell,P.K.
  TITLE     Lack of attenuation of a candidate dengue 1 vaccine (45AZ5) in
            human volunteers
  JOURNAL   Am. J. Trop. Med. Hyg. 36 (2), 435-442 (1987)
   PUBMED   3826504
REFERENCE   3  (bases 1 to 10735)
  CONSRTM   NCBI Genome Project
  TITLE     Direct Submission
  JOURNAL   Submitted (01-AUG-2000) National Center for Biotechnology
            Information, NIH, Bethesda, MD 20894, USA
```

- **To retrieve the DNA sequence for the DEN-1 Dengue virus genome sequence as a FASTA format sequence file, click on “Send” at the top right of the NC_001477 sequence record webpage,**
- **Then choose “File” in the pop-up menu that appears, and then choose FASTA from the “Format” menu that appears, and click on “Create file”**



You can now open the FASTA file containing the DEN-1 Dengue virus genome sequence using WordPad on your computer.



Reading sequence data into R

- **Install seqinr package**

```
if (!requireNamespace("BiocManager", quietly = TRUE))
install.packages("BiocManager")
BiocManager::install("seqinr")
```

- **Load Library**

```
> library("seqinr")
```

- **Read sequence using read.fasta**

```
> dengueseq<- read.fasta(file = "seq.fasta")
```

- **The first element of the list object *dengue* contains the the DNA sequence.**

```
> dengueseq <- dengueseq[[1]]
```

Length of a DNA sequence

- **Once you have retrieved a DNA sequence, we can obtain some simple statistics to describe that sequence, such as the sequence's total length in nucleotides.**

- **To subsequently obtain the length of the genome sequence, we would use the length() function, typing:**

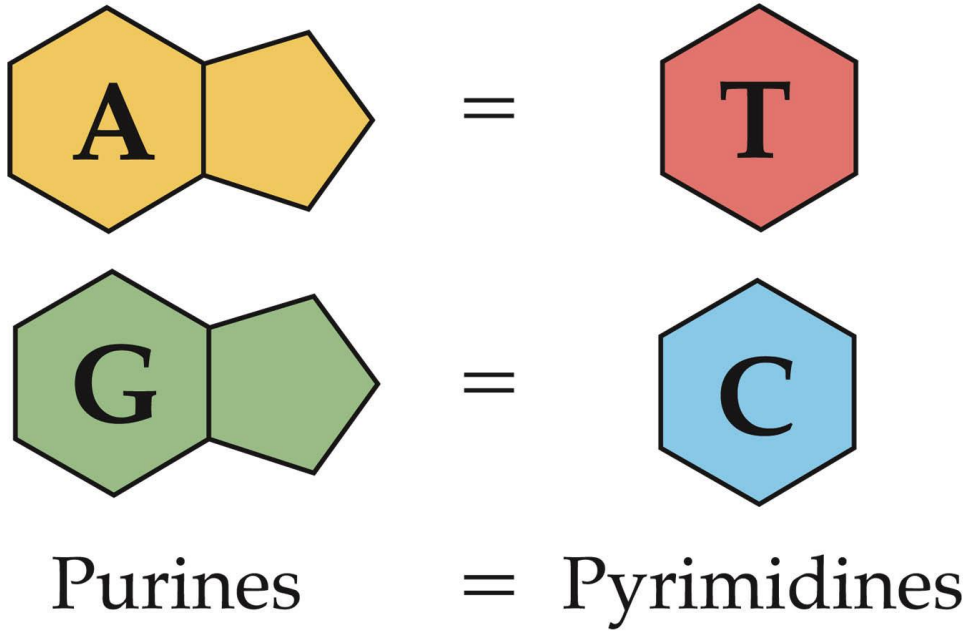
```
> length(dengueseq)
[1] 10735
```

Base composition of a DNA sequence

- **To subsequently obtain the composition of the genome sequence, we would use the table() function, typing:**

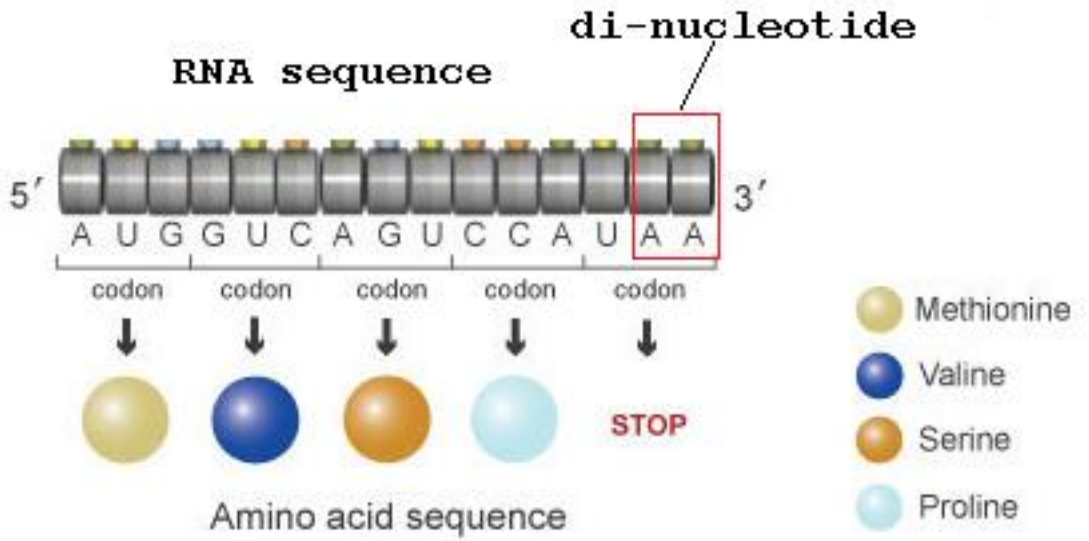
```
> table(dengueseq)

dengueseq
   a    c    g    t
3426 2240 2770 2299
```



Words

- Short strings of letters from an alphabet
- A word of length *k* is called a *k*-word or *k*-tuple
- Examples:
  - 1-tuple: individual nucleotide
  - 2-tuple: dinucleotide
  - 3-tuple: codon



2-words: dinucleotides

- Composed of 2 nucleotides
  - Given DNA alphabet {A,T,C,G}
    - How many possible dinucleoties?
    - Total of 16: AA, AC,AG,AT … TG,TT
- CpG islands are regions of DNA
  - Frequent repetition of CpG dinucleotides
  - Rich in ‘G’ and ‘C’
  - CpG islands appear in some 70% of promoters of human genes

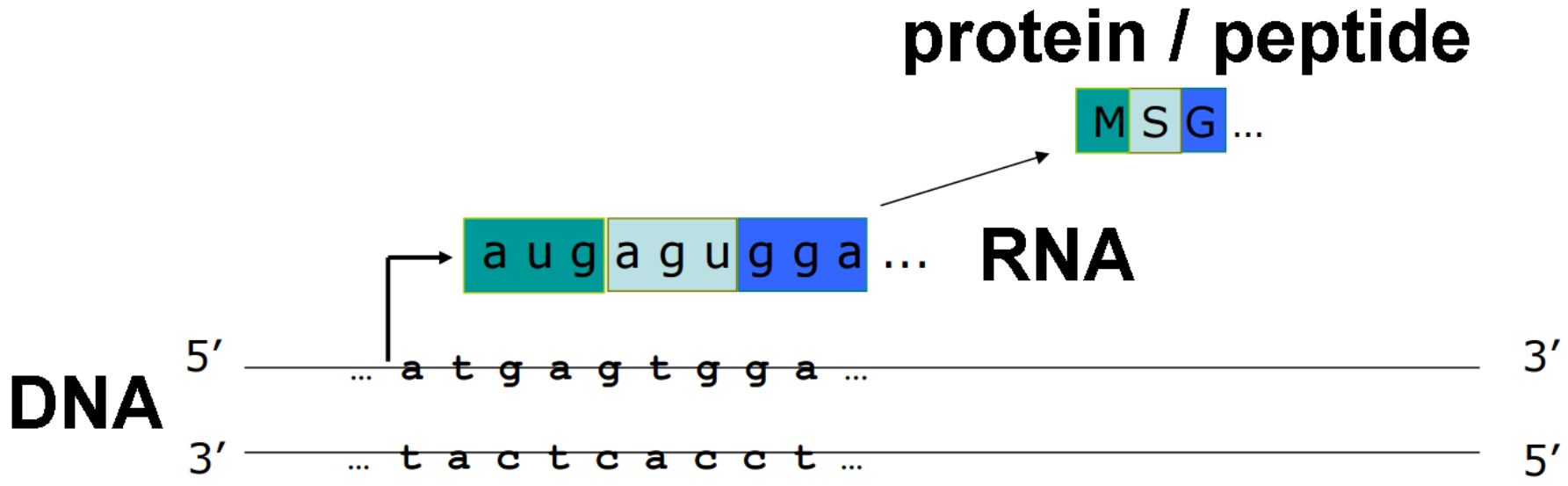
DNA di-nucletides words

- **if you want to know the frequency of all DNA words that are 2 nucleotides long in the Dengue virus genome sequence, you can type:**

```
> count(dengueseq, 2)
 aa  ac  ag  at  ca  cc  cg  ct  ga  gc  gg  gt  ta  tc  tg  tt
1108 720 890 708 901 523 261 555 976 500 787 507 440 497 832 529
```

3-words: codons

- Important in case of DNA sequences
- Linked to expression
  - DNA → RNA → protein



GC Content of DNA

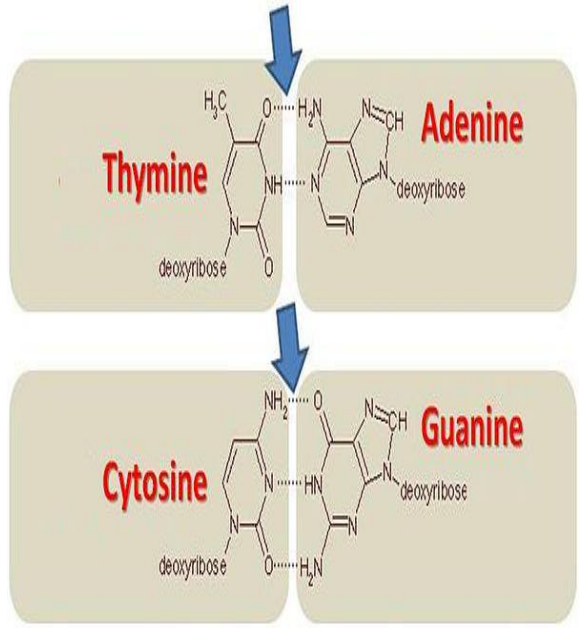
- **One of the most fundamental properties of a genome sequence is its GC content, the fraction of the sequence that consists of Gs and Cs, ie. the %(G+C).**

- **You can easily calculate the GC content based on the number of As, Gs, Cs, and Ts in the genome sequence.**

- **For example, for the DEN-1 Dengue virus genome sequence, we know from using the table() function above that the genome contains 3426 As, 2240 Cs, 2770 Gs and 2299 Ts. Therefore, we can calculate the GC content using the command:**

> GC(dengueseq)

[1] 0.4666977



Nucleotide bonds showing AT and GC pairs. Arrows point to the hydrogen bonds

Local variation in GC content

- **Although the GC content of the whole DEN-1 Dengue virus genome sequence is about 46.7%, there is probably local variation in GC content within the genome.**

- **That is, some regions of the genome sequence may have GC contents quite a bit higher than 46.7%, while some regions of the genome sequence may have GC contents that are quite a big lower than 46.7%.**

- **Local fluctuations in GC content within the genome sequence can provide different interesting information, for example, they may reveal cases of horizontal transfer or reveal biases in mutation.**

A sliding window analysis of GC content

- **In order to study local variation in GC content within a genome sequence, we could calculate the GC content for small chunks of the genome sequence.**

```
> GC(dengueseq[1:2000])      # Calculate the GC content of nucleotides 1-2000 of the Dengue genome
[1] 0.465
> GC(dengueseq[2001:4000])   # Calculate the GC content of nucleotides 2001-4000 of the Dengue genome
[1] 0.4525
> GC(dengueseq[4001:6000])   # Calculate the GC content of nucleotides 4001-6000 of the Dengue genome
[1] 0.4705
> GC(dengueseq[6001:8000])   # Calculate the GC content of nucleotides 6001-8000 of the Dengue genome
[1] 0.479
> GC(dengueseq[8001:10000])  # Calculate the GC content of nucleotides 8001-10000 of the Dengue genome
[1] 0.4545
> GC(dengueseq[10001:10735]) # Calculate the GC content of nucleotides 10001-10735 of the Dengue genome
[1] 0.4993197
```

for loop in R

- **In R, it is possible to write a *for loop* to carry out the same command several times.**

- **For example, if we want to print out the square of each number between 1 and 10, we can write the following for loop:**

```
> for (i in 1:10) { print (i*i) }
[1] 1
[1] 4
[1] 9
[1] 16
[1] 25
[1] 36
[1] 49
[1] 64
[1] 81
[1] 100
```

the variable *i* is a counter for the number of cycles through the loop

In the first cycle through the loop, the value of *i* is 1, and so *i* * *i* =1 is printed out.

In the second cycle through the loop, the value of *i* is 2, and so *i* * *i* =4 is printed out

Lets us create a new function

- **We can also create our own functions in R to do calculations that you want to carry out very often on different input data sets.**

- **For example, we can create a function to calculate the value of 20 plus the square of some input number:**

```
> myfunction <- function(x) { return(20 + (x*x)) }
```

This function will calculate the square of a number (*x*), and then add 20 to that value. The return() statement returns the calculated value.

- **we can use the function for different input numbers (eg. 10, 25):**

```
> myfunction(10)
[1] 120
> myfunction(25)
[1] 645
```

For loop - GC content

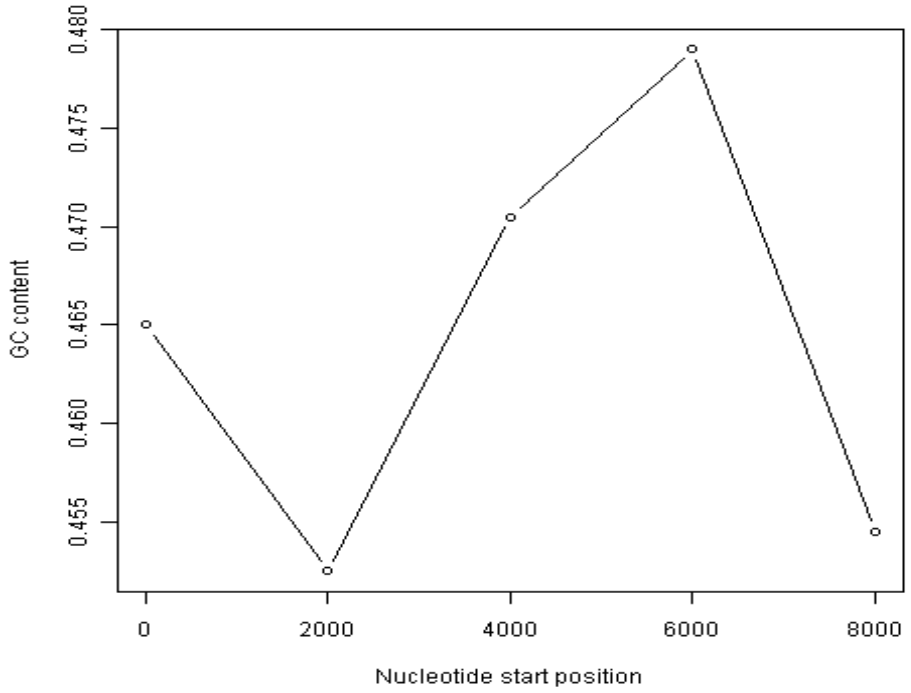
```
> starts <- seq(1, length(dengueseq)-2000, by = 2000)
> n <- length(starts)    # Find the length of the vector
"starts"
> chunkGCs <- numeric(n) # Make a vector of the same
length as vector "starts", but just containing zeroes
> for (i in 1:n) {
      chunk <- dengueseq[starts[i]:(starts[i]+1999)]
      chunkGC <- GC(chunk)
      print(chunkGC)
      chunkGCs[i] <- chunkGC
    }
> plot(starts,chunkGCs,type="b",xlab="Nucleotide start
position",ylab="GC content")
```

We set the variable *n* to be equal to the number of elements in the vector *starts*,

The line “for (i in 1:n)” means that the counter *i* will take values of 1-5 in subsequent cycles of the *for loop*.

A sliding window plot of GC content

```
> plot(starts,chunkGC,type="b",xlab="Nucleotide start position",ylab="GC content")
```

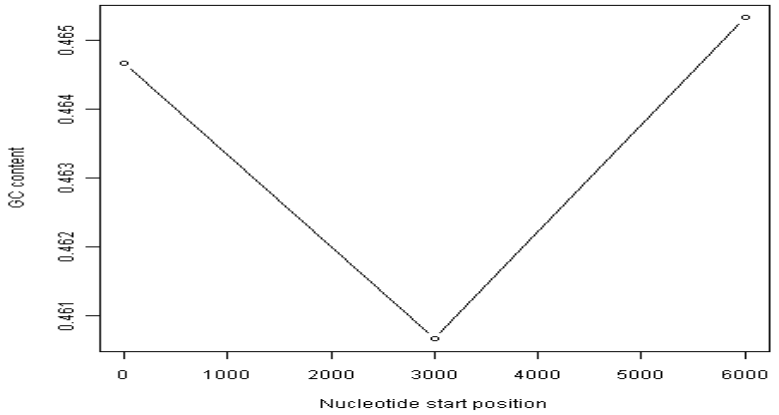


Create a new Function to plot slidingwindoplot

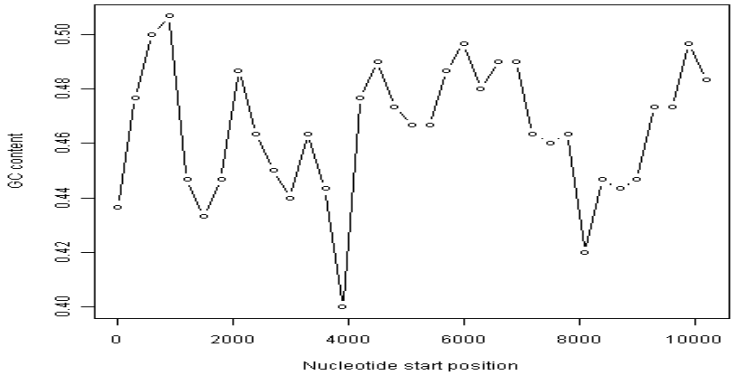
```
> slidingwindowplot <- function(windowsize, inputseq)
{
  starts <- seq(1, length(inputseq)-windowsize, by = windowsize)
  n <- length(starts)                                  # Find the length of the vector "starts"
  chunkGCs <- numeric(n)   # Make a vector of the same length as vector "starts", but just containing zeroes
  for (i in 1:n) {
      chunk <- inputseq[starts[i]:(starts[i]+windowsize-1)]
      chunkGC <- GC(chunk)
      print(chunkGC)
      chunkGCs[i] <- chunkGC
  }
  plot(starts,chunkGCs,type="b",xlab="Nucleotide start position",ylab="GC content")
}
```

Let us plot GC content in different window size

```
> slidingwindowplot(3000, dengueseq)
```




```
> slidingwindowplot(300, dengueseq)
```



Over and under represented words (1)

- **It is interesting to identify DNA words that are two nucleotides long (“dinucleotides”, ie. “AT”, “AC”, etc.) that are over-represented or under-represented in a DNA sequence.**

- **If a particular DNA word is *over-represented* in a sequence, it means that it occurs many more times in the sequence than you would have expected by chance.**

- **Similarly, if a particular DNA word is *under-represented* in a sequence, it means it occurs far fewer times in the sequence than you would have expected.**

Over and under represented words (2)

- A statistic called ρ (Rho) is used to measure how over- or under-represented a particular DNA word is. For a 2-nucleotide (dinucleotide) DNA word ρ is calculated as:

$$\rho(xy) = f_{xy}/(f_x * f_y),$$

where f_{xy} and f_x are the frequencies of the DNA words *xy* and *x* in the DNA sequence under study.

- For example, the value of ρ for the DNA word "TA" can be calculated as: $\rho(TA) = f_{TA}/(f_T * f_A)$, where f_{TA}, f_T and f_A are the frequencies of the DNA words "TA", "T" and "A" in the DNA sequence.

Over and under represented words (3)

- **The frequencies of the 2-nucleotide DNA words in a sequence are expected to be equal the products of the specific frequencies of the two nucleotides that compose them.**

- **If this were true, then ρ would be equal to 1.**

- **If we find that ρ is much greater than 1 for a particular 2-nucleotide word in a sequence, it indicates that that 2-nucleotide word is much more common in that sequence than expected (ie. it is *over-represented*).**

- **If ρ is much less than 1 , for a particular 2-nucleotide word in a sequence, indicates under represented**

Let us calculate Rho (ρ) for GC

```
> count(dengueseq, 1) # Get the number of occurrences of 1-nucleotide DNA words
   a    c    g    t
3426 2240 2770 2299
> 2770/(3426+2240+2770+2299) # Get fG
[1] 0.2580345
> 2240/(3426+2240+2770+2299) # Get fC
[1] 0.2086633
> count(dengueseq, 2) # Get the number of occurrences of 2-nucleotide DNA words
  aa  ac  ag  at  ca  cc  cg  ct  ga  gc  gg  gt  ta  tc  tg  tt
1108 720 890 708 901 523 261 555 976 500 787 507 440 497 832 529
> 500/(1108+720+890+708+901+523+261+555+976+500+787+507+440+497+832+529)
# Get fGC
[1] 0.04658096
> 0.04658096/(0.2580345*0.2086633) # Get rho(GC)
[1] 0.8651364
```

Exercise

Check how many of these are over and under represented sequences in dengu sequence

- **TA**
- **GA**
- **CT**

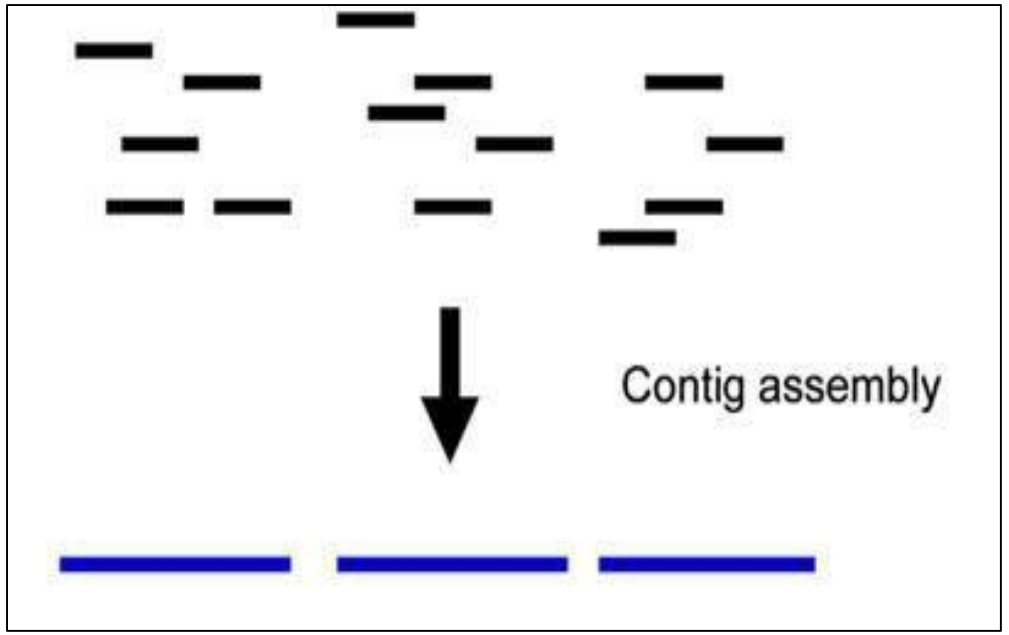
What is Sequence Alignment ?

A sequence alignment is a way of arranging the sequences of DNA , RNA, or protein to identify regions of similarity.

Comparable ?

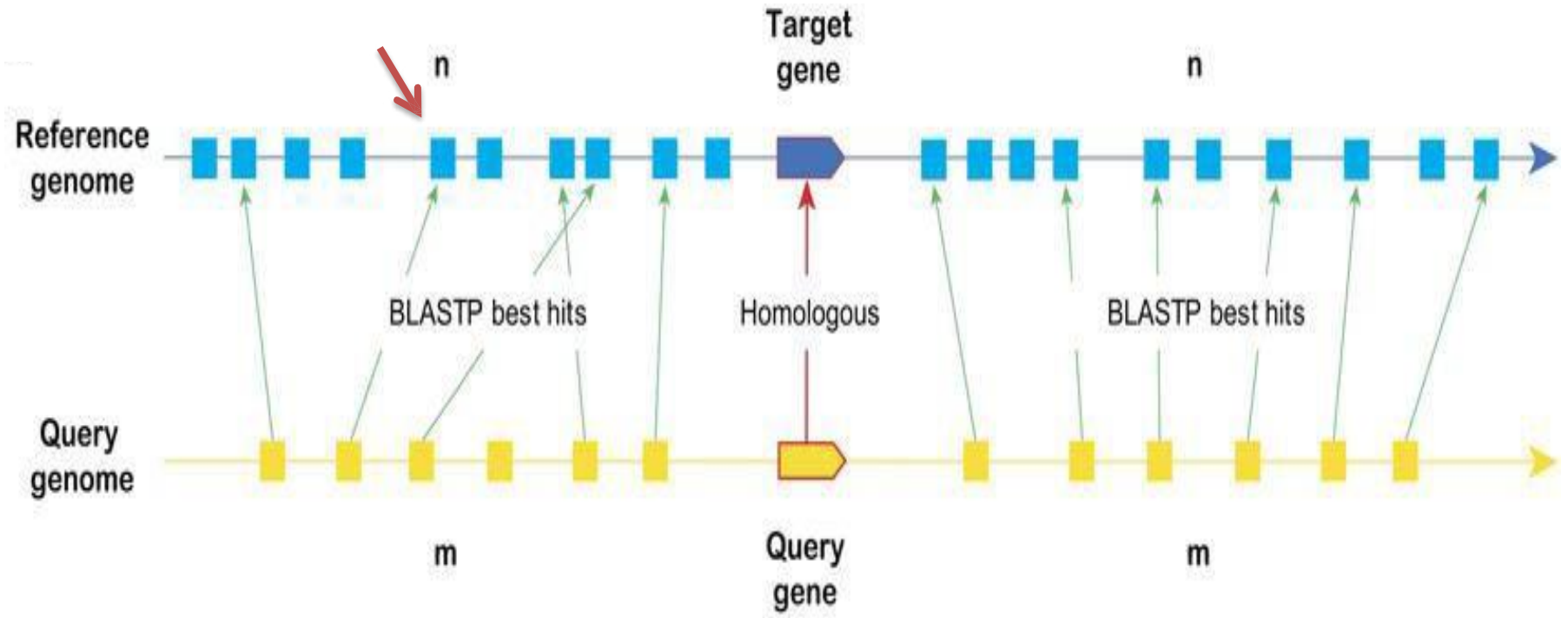
Sequence Alignment :Uses (1)

- **Sequence Assembly : Genome sequence are assembled by using the sequence alignment methods to find the overlap between many short pieces of DNA .**



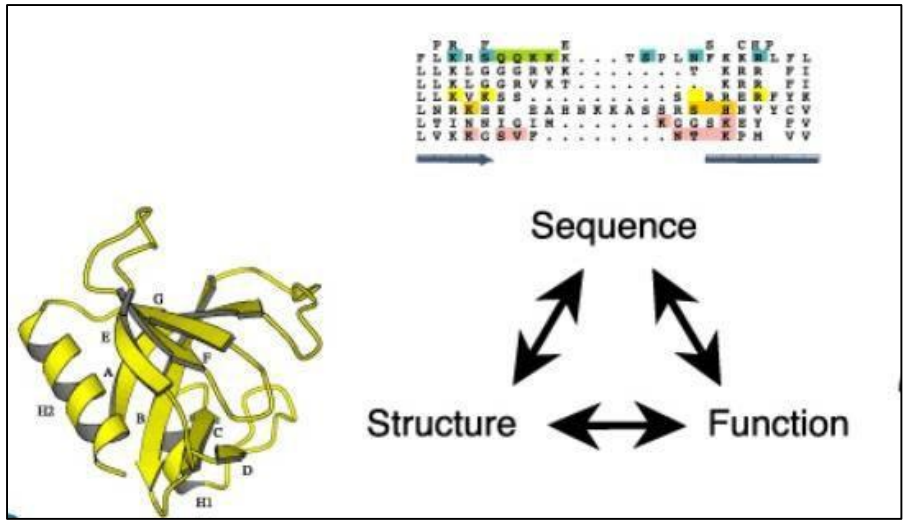
Sequence Alignment :Uses (2)

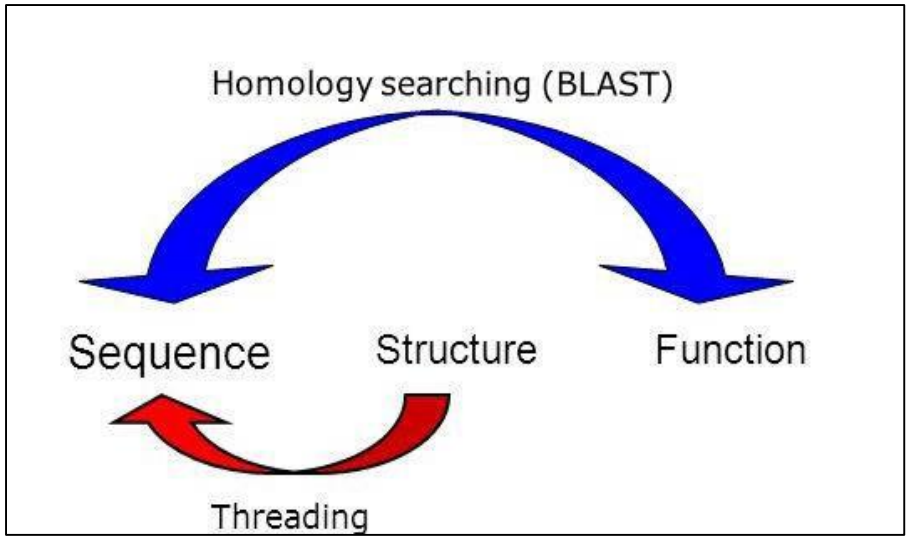
- **Gene Finding : Sequence similarity could help us to find the gene prediction just by doing comparison against the other set of sequences.**



Sequence Alignment :Uses (3)

- **Function prediction : Function of any unknown sequence could be predicted by comparing with other known sequence .**

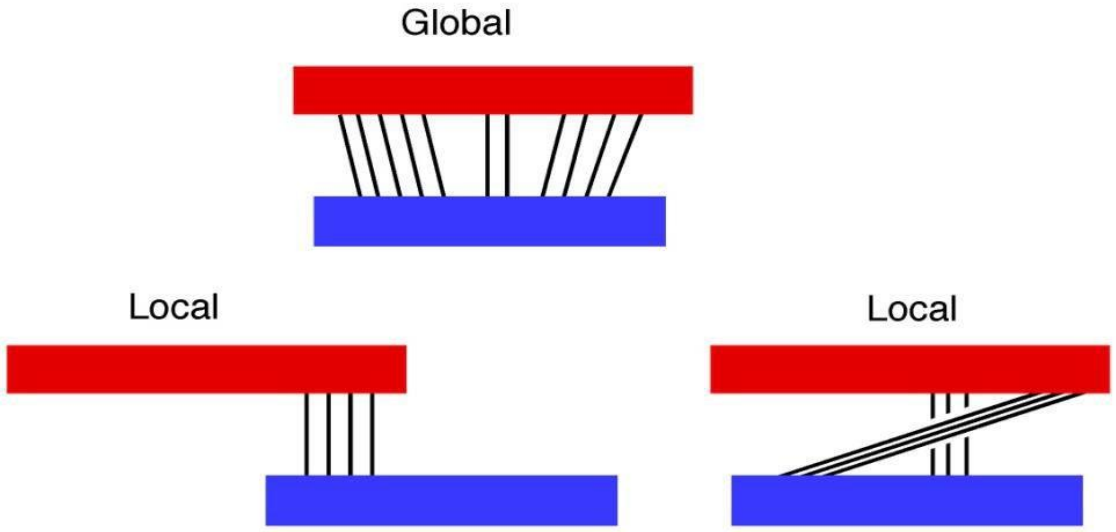




Types of Alignments

- **Global : This attempt to align every residue in every sequence.**

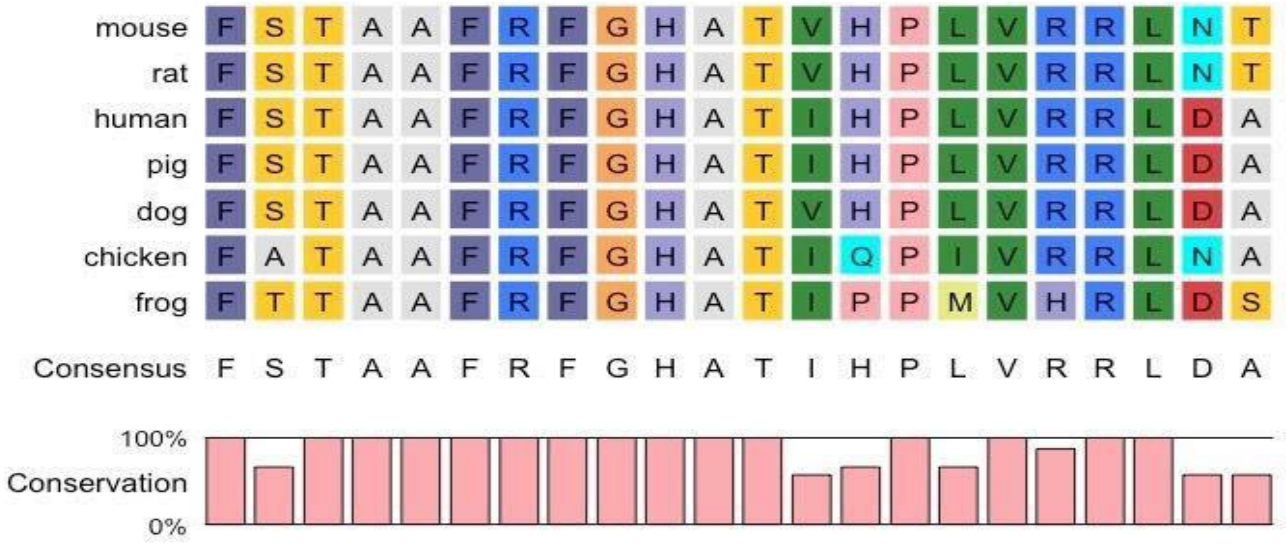
- **Local: It is more useful for dissimilar sequences that are suspected to contain regions of similarity or similar sequence motifs within their larger sequence context.**



Types of Alignments: Based on number of sequences

- **Pair wise Sequence Alignment** : This alignments can only be used between two sequences at a time.

- **Multiple Sequence Alignment** : This alignments can only be used between more than two sequences at a time.



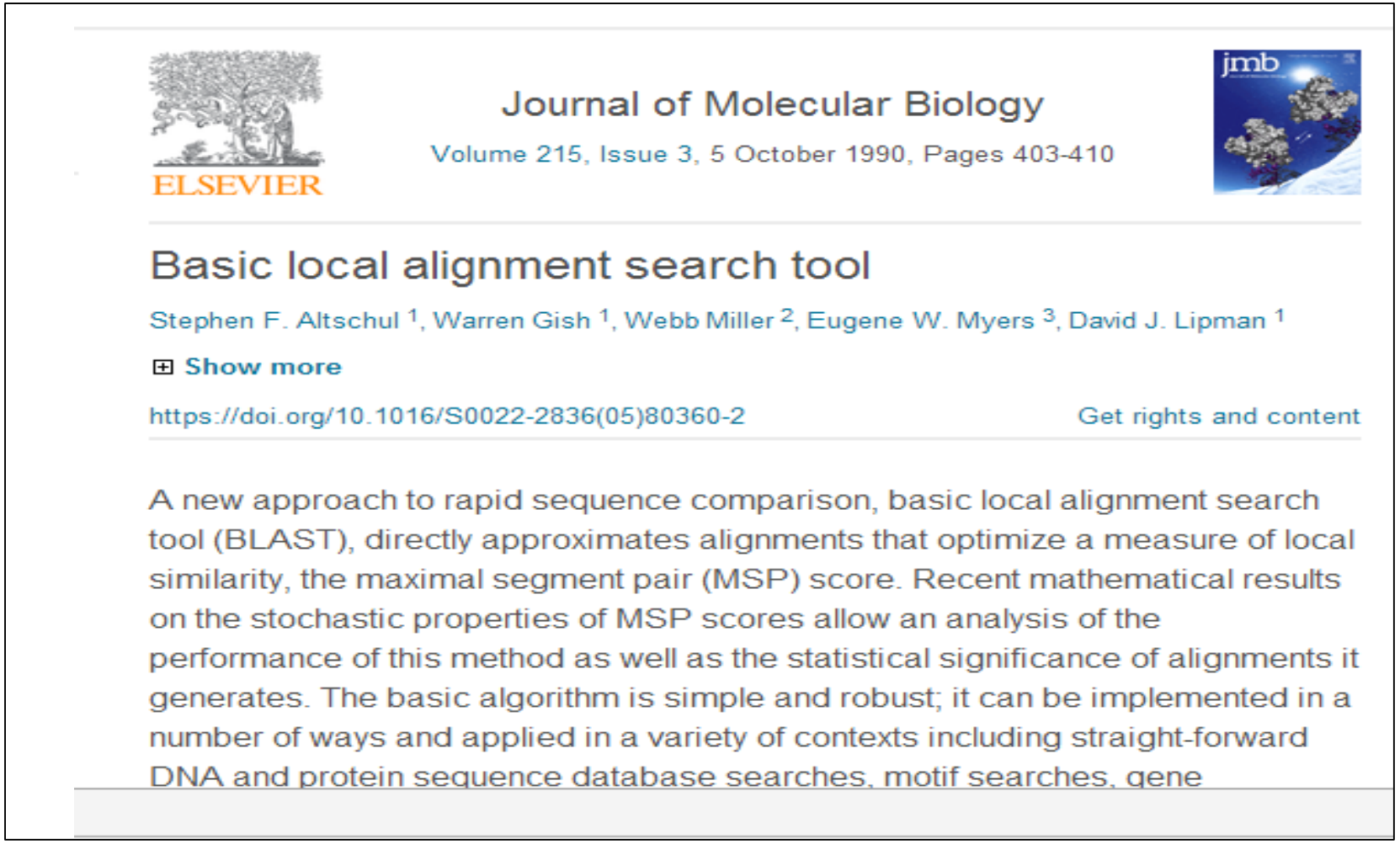
Tools for Sequence Alignments

There are many tools for sequence Alignment. In this session, we will discuss about

- **BLAST**
- **CLUSTALW**

Sequence Alignment : BLAST

- **BLAST** stands for **B**asic **L**ocal **A**lignment **S**earch **T**ool



ELSEVIER

Journal of Molecular Biology

Volume 215, Issue 3, 5 October 1990, Pages 403-410

Basic local alignment search tool

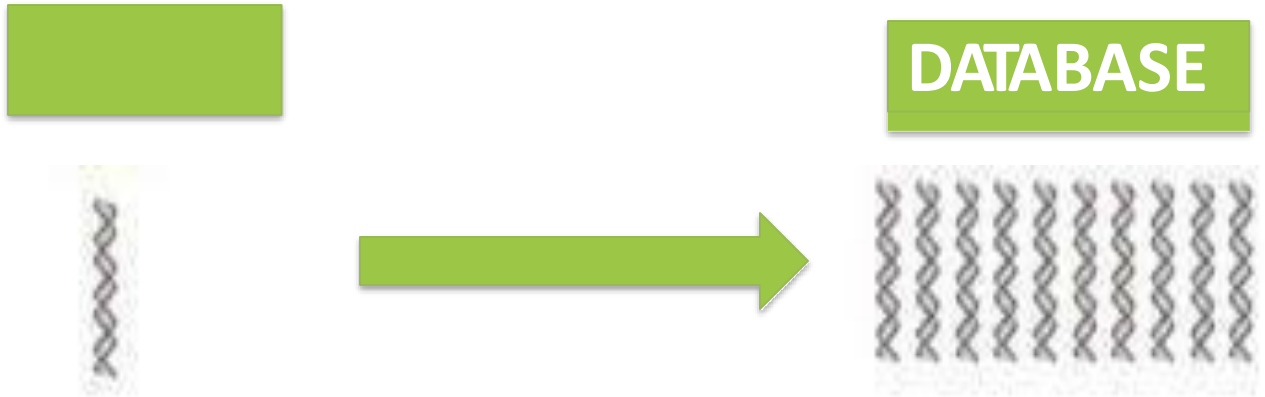
Stephen F. Altschul [1], Warren Gish [1], Webb Miller [2], Eugene W. Myers [3], David J. Lipman [1]

Show more

https://doi.org/10.1016/S0022-2836(05)80360-2 Get rights and content

A new approach to rapid sequence comparison, basic local alignment search tool (BLAST), directly approximates alignments that optimize a measure of local similarity, the maximal segment pair (MSP) score. Recent mathematical results on the stochastic properties of MSP scores allow an analysis of the performance of this method as well as the statistical significance of alignments it generates. The basic algorithm is simple and robust; it can be implemented in a number of ways and applied in a variety of contexts including straight-forward DNA and protein sequence database searches, motif searches, gene

- **A BLAST search enables a researcher to compare a query sequence with a library or databases of sequences, and identify library sequences that resemble the query sequence above a certain threshold.**



Types of BLAST (1)

- **BLASTN**

nucleotide query : search nucleotide databases using a

(A)Query : ATGCATCGATC

(B) Database : ATCGATGATCGACATCGATCAGCTACG

- **BLASTP : search protein databases using a protein query**

(A)Query : VIVALASVEGAS

(B) DATABASE : TARDEFGGAVIVADAVISASTILHGGQWLC

- **BLASTX : search protein databases using a translated nucleotide query**

(A)Query : ATGCATCGATC

(B)DATABASE : TARDEFGGAVIVADAVISASTILHGGQWLC

Types of BLAST (2)

- **TBLASTN : search translated nucleotide databases using a protein query**

(A)Query : TARDEFGGAVI
(B)DATABASE : ATCGATGATCGACATCGATCAGCTACG

- **TBLASTX : search translated nucleotide databases using a translated nucleotide query**

(A)Query : CGATGATCG
(B)DATABASE : ATCGATGATCGACATCGATCAGCTACG

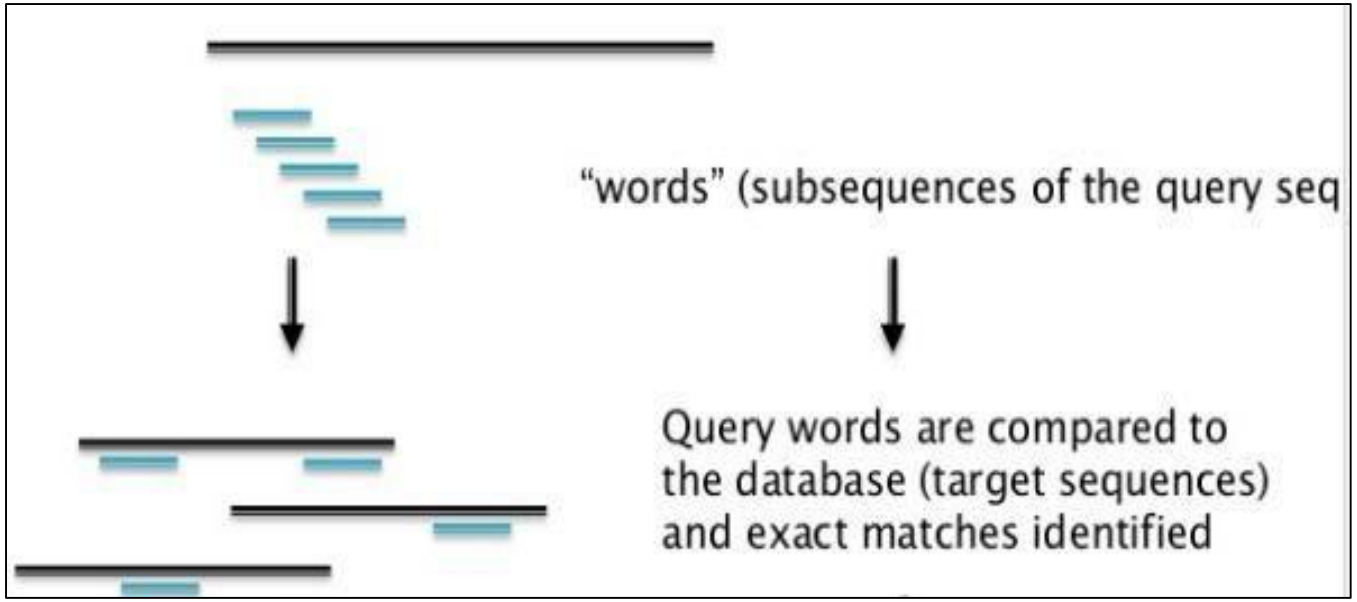
Types of BLAST : ALL

Program	Database	Query
BLASTN	Nucleotide	Nucleotide
BLASTP	Protein	Protein
BLASTX	Protein	Nt. → Protein
TBLASTN	Nt. → Protein	Protein
TBLASTX	Nt. → Protein	Nt. → Protein

How does BLAST Works?

- **Construct a dictionary of all words in the query**

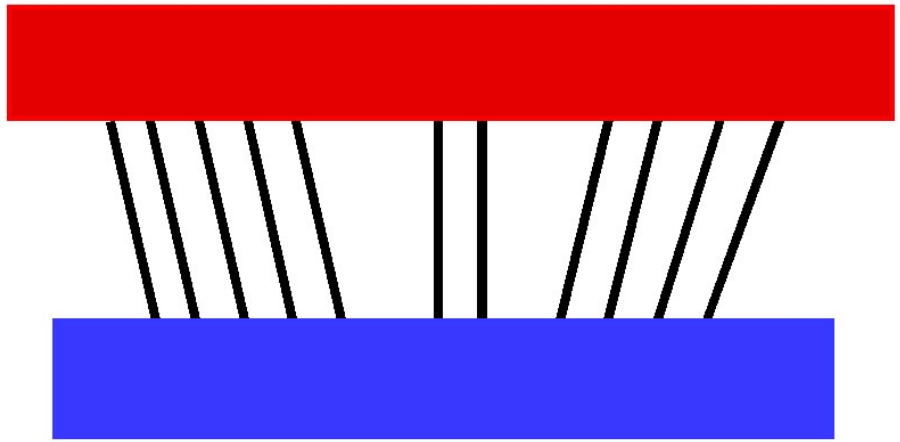
- **Initiate a local alignment for each word match between query and DB**



BLAST: Global Alignment

- It compares the whole sequence with another sequence.
- So, output of Global is one to one comparison of two sequences.
- This method is useful if you have small group of sequences.

Global alignment



Global alignment (NW -Needleman–Wunsch)

- **Sequences are aligned end-to-end along their entire length**
- **Many possible alignments are produced**
  - **The alignment with the highest score is chosen**
- **Naïve algorithm is very inefficient (O^{exp})**
  - **Impractical for sequences of length >20 nt**
- **Used to analyze homology/similarity of**
  - **genes and proteins**
  - **between species**

Methodology of global alignment (1 of 4)

- Define scoring scheme for each event
  - mismatch between a_i and b_j
    - $s(a_i, b_j) = -1$ if $a_i \neq b_j$
  - gap (insertion or deletion)
    - $s(a_i, -) = s(-, b_j) = -2$
  - match between a_i and b_j
    - $s(a_i, b_j) = +2$ if $a_i = b_j$
- Provide no restrictions on minimal score
- Start completing the alignment MxN matrix

```
s1: ..AATA..
s2: ..AACA..

s1: ..AAT-A..
s2: ..AACA..

s1: ..AATA..
s2: ..AATA..
```

Methodology of global alignment (2 of 4)

- The matrix should have **extra** column and row
  - **M+1** columns , where M is the length sequence M
  - **N+1** rows, where N is the length of sequence N

1. Initialize the matrix
   - introduce **gap penalty** at every **initial** position along rows and columns
   - Scores at each cell are **cumulative**

		W	H	A	T
	0 -2→	-2 -2→	-4 -2→	-6 -2→	-8
W	-2 -2↓				
H	-4 -2↓				
Y	-6				

Methodology of global alignment (3 of 4)

2. Alignment possibilities

Gap (horiz/vert)

		W	H
	0	-2	-4
W	-2	-4	

-2 -2

-OR-

Match (W-W diag.)

		W	H
	0	-2	-4
W	-2	+2	

+2

3. Select **the maximum** score

- Best alignment

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

Methodology of global alignment (4 of 4)

4. Select the most **very bottom right** cell
5. Consider different path(s) going to **very top left cell**
   - How the next cell value was generated? From where?

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

WHAT
WHY–

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

WHAT
WH–Y

Exercise

1. Creat scoring matrix of CTTCA and CTACA
   where mismatch -1,
   gap -2 and
   Match +5

2. Give the track back arrow

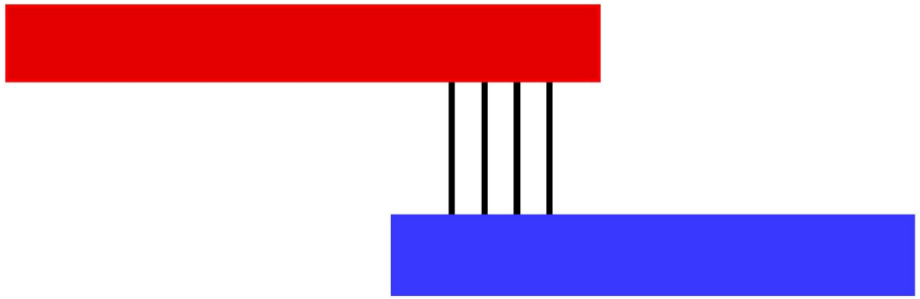
3. Write alignment

BLAST: Local Alignment

- Local method uses the subset of sequence and attempts to align against the subset of another sequence.
- So, output of local alignment gives the subset of regions which are highly similar.
- Example : Compare two sequence A and B

(A) GCATTACTAATATATTAGTAAATCAGAGTAGTA
|||||||||
(B) AAGCGAATAATATATTTATACTCAGATTATTGCGCG

Local alignment



Local alignment (Smith–Waterman)

- Sequences are aligned to find regions where the best alignment occurs (i.e. highest score)
- Assumes a local context (aligning parts of seq.)
- Ideal for finding short motifs, DNA binding sites
  - helix-loop-helix (bHLH) - motif
  - TATAAT box (a famous promoter region) – DNA binding site
- Works well on highly divergent sequences

Methodology of local alignment (1 of 4)

- The scoring system is similar with one exception
  - The minimum possible score in the matrix is zero
  - There are no negative scores in the matrix
- Let's define the scoring system as in global

mismatch between seq. a_i and b_j

$s(a_i, b_j) = -1$ if $a_i \neq b_j$

gap (insertion or deletion)

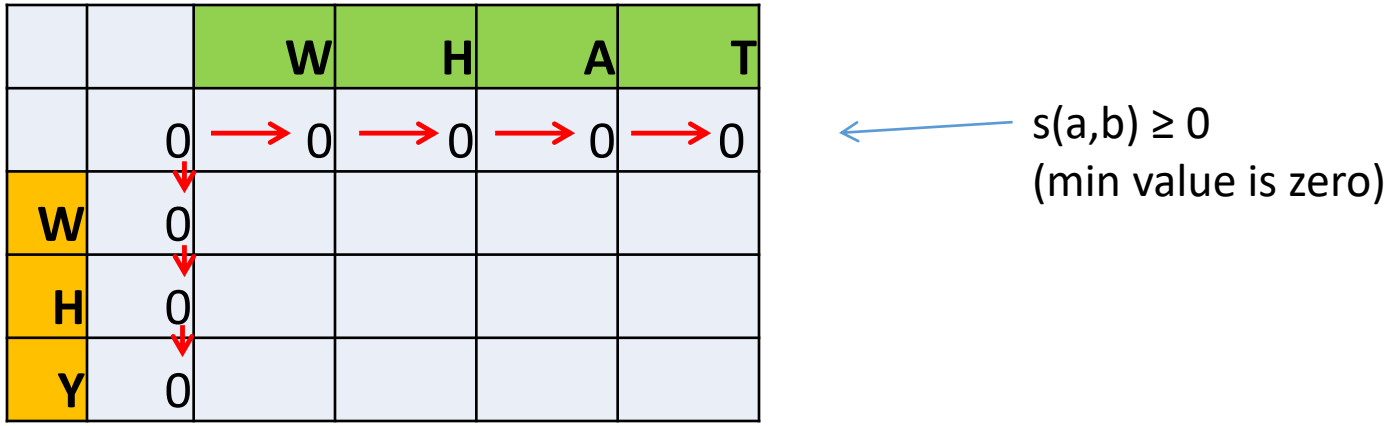
$s(a_i, -) = s(-, b_j) = -2$

match between a_i and b_j

$s(a_i, b_j) = +2$ if $a_i = b_j$

Methodology of local alignment (2 of 4)

- **Construct the MxN alignment matrix with M+1 columns and N+1 rows**
- **Initialize the matrix by introducing gap penalty at 1st row and 1st column**



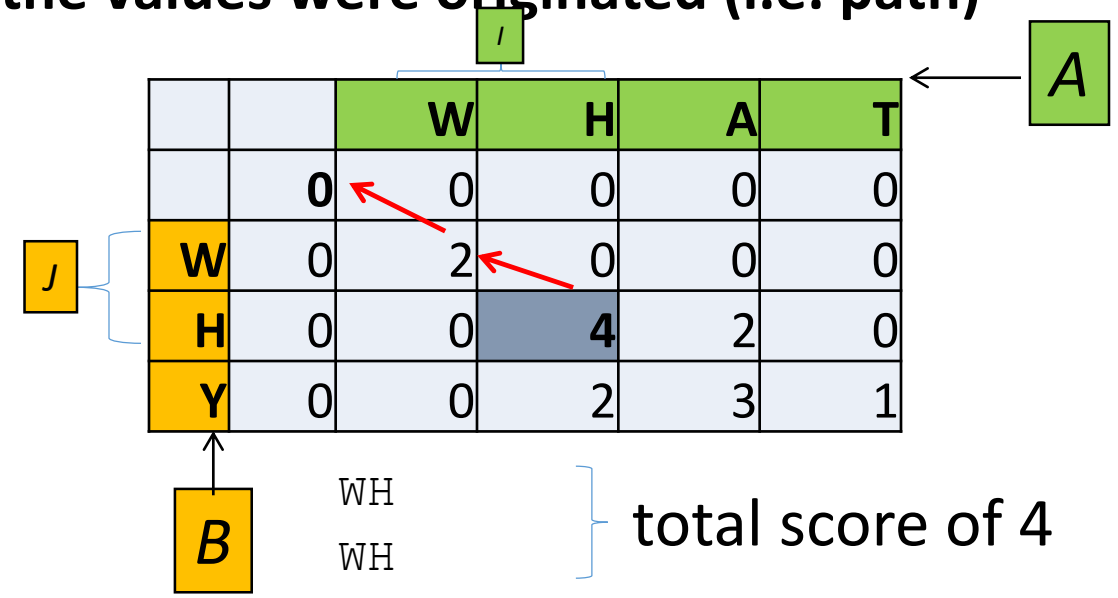
Methodology of local alignment (3 of 4)

- For each subsequent cell consider alignments
  - Vertical *s(I, -)*
  - Horizontal *s(-,J)*
  - Diagonal *s(I,J)*
- For each cell select the highest score
  - If score is negative → assign zero

		W	H	A	T
	0	0	0	0	0
W	0	2	0	0	0
H	0	0	4	2	0
Y	0	0	2	3	1

Methodology of local alignment (4 of 4)

- Select the initial cell with the highest score(s)
- Consider different path(s) leading to score of **zero**
  - **Trace-back the cell values**
  - **Look how the values were originated (i.e. path)**



- Mathematically $M(A,B) = \max\{S(I,J) : I \subset A, J \subset B\}$
  - where *S*(*I*, *J*) is the score for **sub-sequences** *I* and *J*

Local alignment illustration (1 of 2)

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

- **Sequence A:** ACCTAAGG
- **Sequence B:** GGCTCAATCA
- Scoring conditions:
  - $s(a_i, b_j) = +2$ if $a_i = b_j$,
  - $s(a_i, b_j) = -1$ if $a_i \neq b_j$ and
  - $s(a_i, -) = s(-, b_j) = -2$

Local alignment illustration (2 of 2)

		G	G	C	T	C	A	A	T	C	A
	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	0	2	2	0	0	2
C	0	0	0	2	0	2	0	1	1	2	0
C	0	0	0	2	1	2	1	0	0	3	1
T	0	0	0	0	4	2	1	0	2	1	2
A	0	0	0	0	2	3	4	3	1	1	3
A	0	0	0	0	0	1	5	6	4	2	3
G	0	2	2	0	0	0	3	4	5	3	1
G	0	2	4	2	0	0	1	2	3	4	2

Local alignment illustration (3 of 3)

		G	G	C	T	C	A	A	T	C	A
	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	0	2	2	0	0	2
C	0	0	0	2	0	2	0	1	1	2	0
C	0	0	0	2	1	2	1	0	0	3	1
T	0	0	0	0	4	2	1	0	2	1	1
A	0	0	0	0	2	3	4	3	1	1	3
A	0	0	0	0	0	1	5	6	4	2	3
G	0	2	2	0	0	0	3	4	5	3	1
G	0	2	4	2	0	0	1	2	3	4	2

CTCAA

CT−AA

locally

BLAST: Input Format

Many program for sequence alignment expect sequences to be in FASTA format

Example 1 :

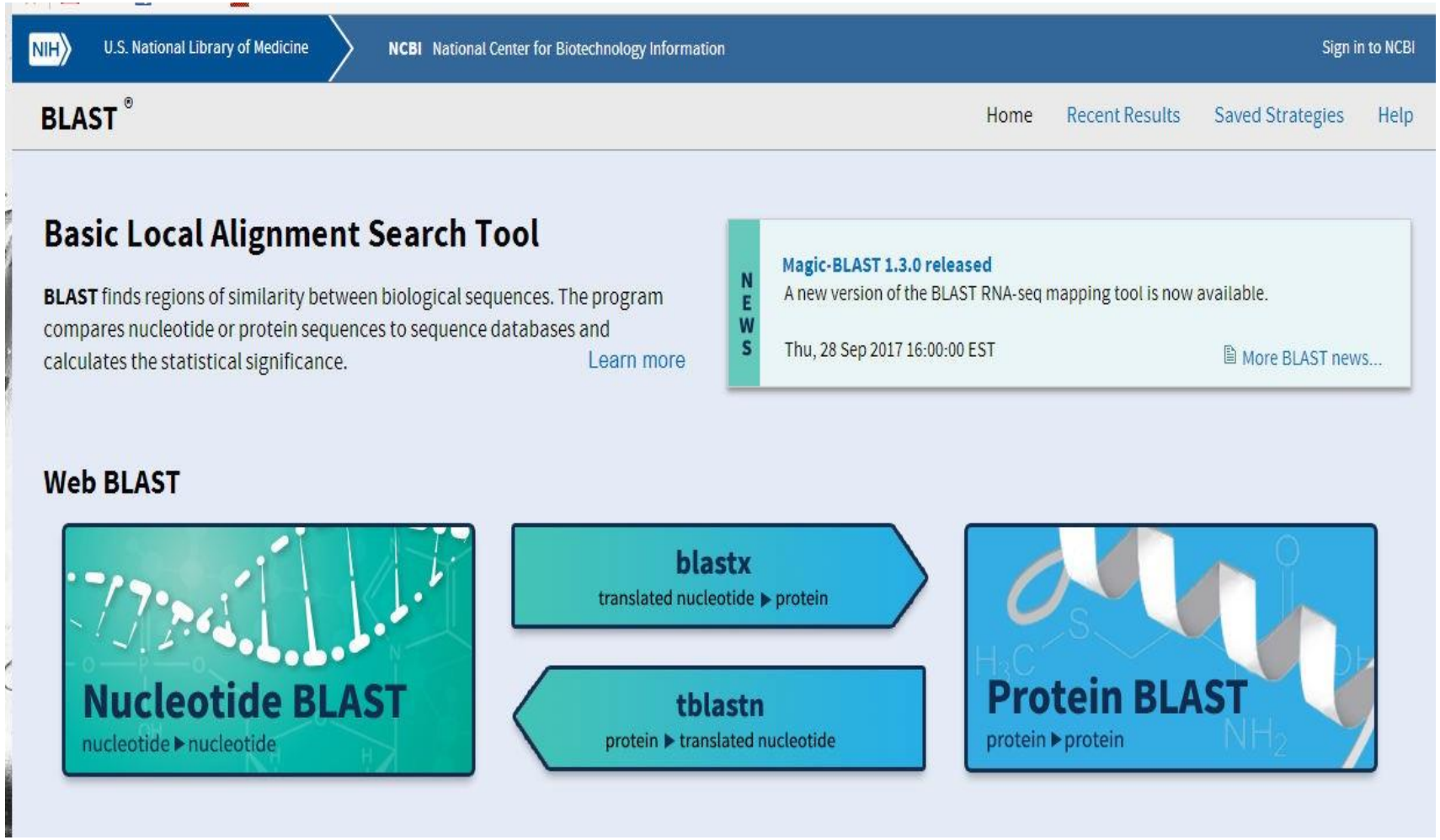
```
>L37107.1 Canis familiaris p53 mRNA, partial cds
GTTCCGTTTGGGGTTCCTGCATTCCGGGACAGCCAAGTCTGTTACTTGGACGTACTCCCCTCTCCTCAAC
AAGTTGTTTTGCCAGCTGGCGAAGACCTGCCCCGTGCAGCTGTGGGTCAGCTCCCCACCCCCACCCAATA
CCTGCGTCCGCGCTATGGCCATCTATAAGAAGTCGGAGTTCGTGACCGAGGTTGTGCGGCGCTGCCCCCA
CCATGAACGCTGCTCTGACAGTAGTGACGGTCTTGCCCCTCCTCAGCATCTCATCCGAGTGGAAGGAAAT
TTGCGGGCCAAGTACCTGGACGACAGAAACACTTTTCGACACAGTGTGGTGGTGCCTTATGAGCCACCCG
AGGTTGGCTCTGACTATACCACCATCCACTACAACTACATGTGTAACAGTTCCTGCATGGGAGGCATGAA
CCGGCGGCCCATCCTCACTATCATCACCCTGGAAGACTCCAGTGGAAACGTGCTGGGACGCAACAGCTTT
GAGGTACGCGTTTGTGCCTGTCCCGGGAGAGACCGCCGGACTGAGGAGGAGAATTTCCACAAGAAGGGGG
AGCCTTGTCCTGAGCCACCCCCCGGGAGTACCAAGCGAGCACTGCCTCCCAGCACCAGCTCCTCTCCCCC
GCAAAAGAAGAAGCCACTAGATGGAGAATATTTCACCCTTCAGATCCGTGGGCGTGAACGCTATGAGATG
TTCAGGAATCTGAATGAAGCCTTGGAGCTGAAGGATGCCCAGAGTGGAAAGGAGCCAGGGGGAAGCAGGG
CTCACTCCAGCCACCTGAAGGCAAAGAAGGGGCAATCTACCTCTCGCCATAAAAAACTGATGTTCAAGAGAG
AA
```

Example 2 :

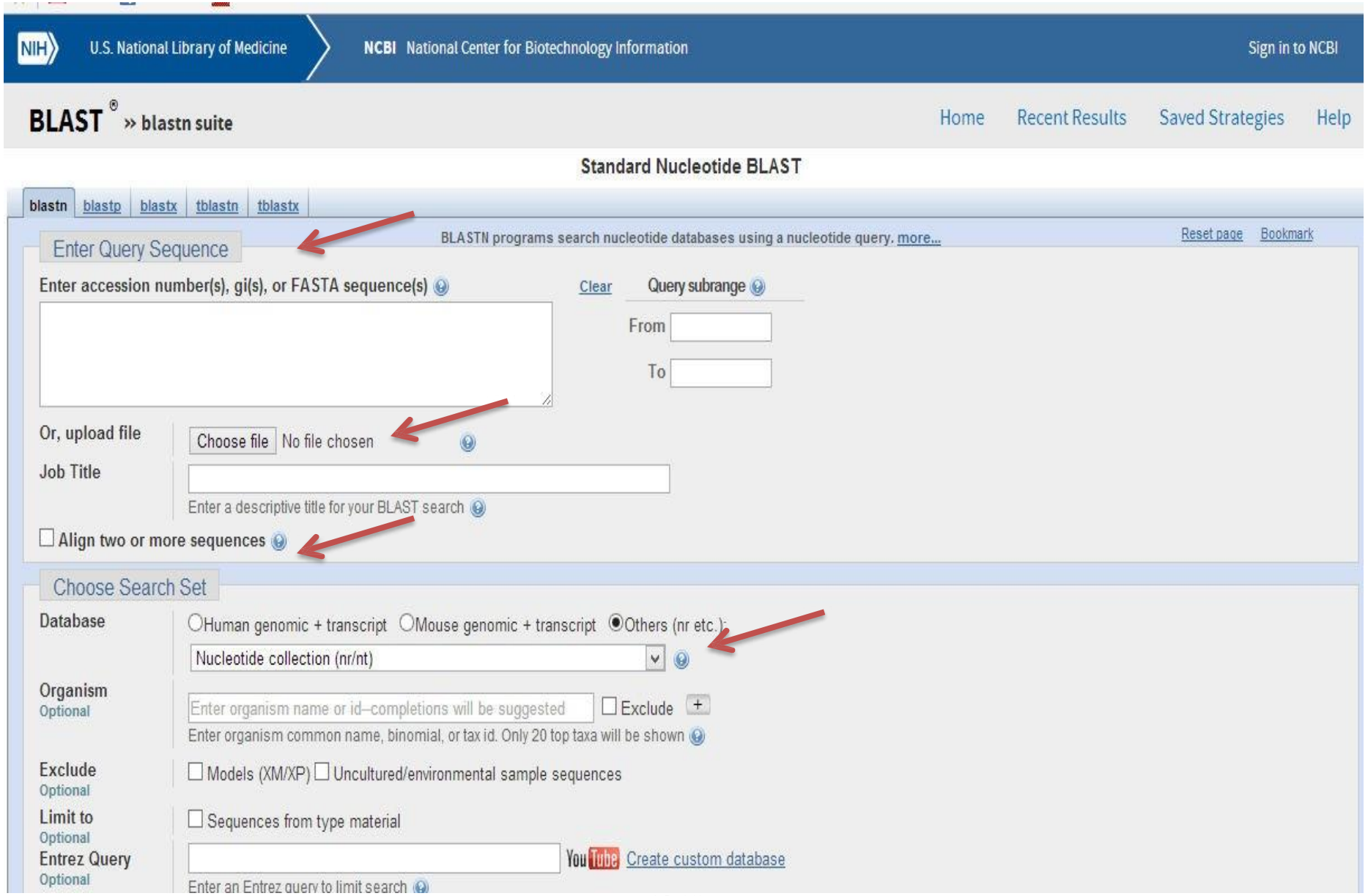
```
>NM_033360.3 Homo sapiens KRAS proto-oncogene, GTPase (KRAS), transcript
variant a, mRNA
TCCTAGGCGGCGGCCGCGGCGGCGGAGGCAGCAGCGGCGGCGGCAGTGGCGGCGGCGAAGGTGGCGGCGG
CTCGGCCAGTACTCCCGGCCCCCGCCATTTCGGACTGGGAGCGAGCGCGGCGCAGGCACTGAAGGCGGCG
GCGGGGCCAGAGGCTCAGCGGCTCCCAGGTGCGGGAGAGAGGCCTGCTGAAAATGACTGAATATAAACTT
GTGGTAGTTGGAGCTGGTGGCGTAGGCAAGAGTGCCTTGACGATACAGCTAATTCAGAATCATTTTGTGG
ACGAATATGATCCAACAATAGAGGATTCCTACAGGAAGCAAGTAGTAATTGATGGAGAAACCTGTCTCTT
GATATTCTCGACACAGCAGGTCAAGAGGAGTACAGTGCAATGAGGGACCAGTACATGAGGACTGGGGAG
GGCTTTCTTTGTGTATTTGCCATAAATAATACTAAATCATTTGAAGATATTCACCATTATAGAGAACAAA
```

NCBI BLAST SERVER

Open the website : https://blast.ncbi.nlm.nih.gov/Blast.cgi



Window of BLASTN



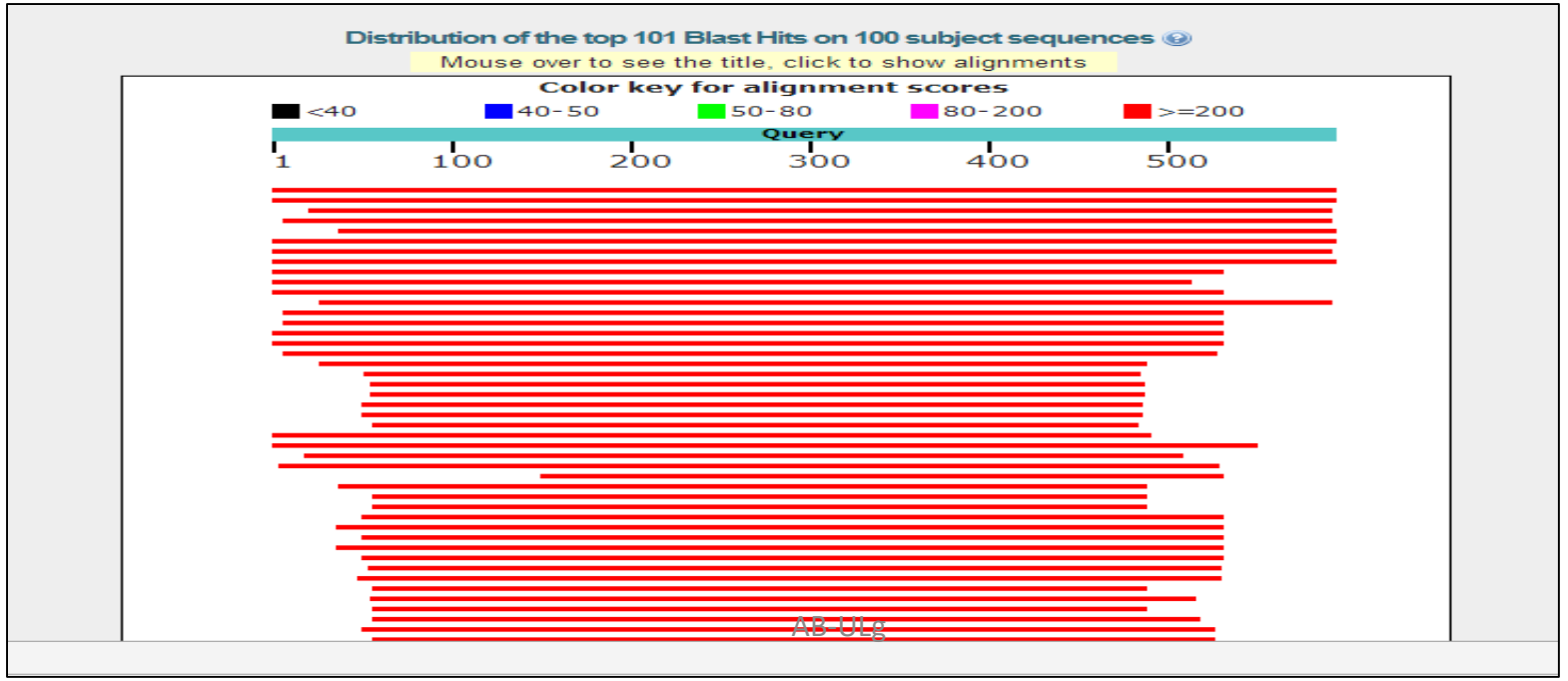
Let us work on BLASTN

- **Select following sequence and give input into NCBI BLASTN query section**

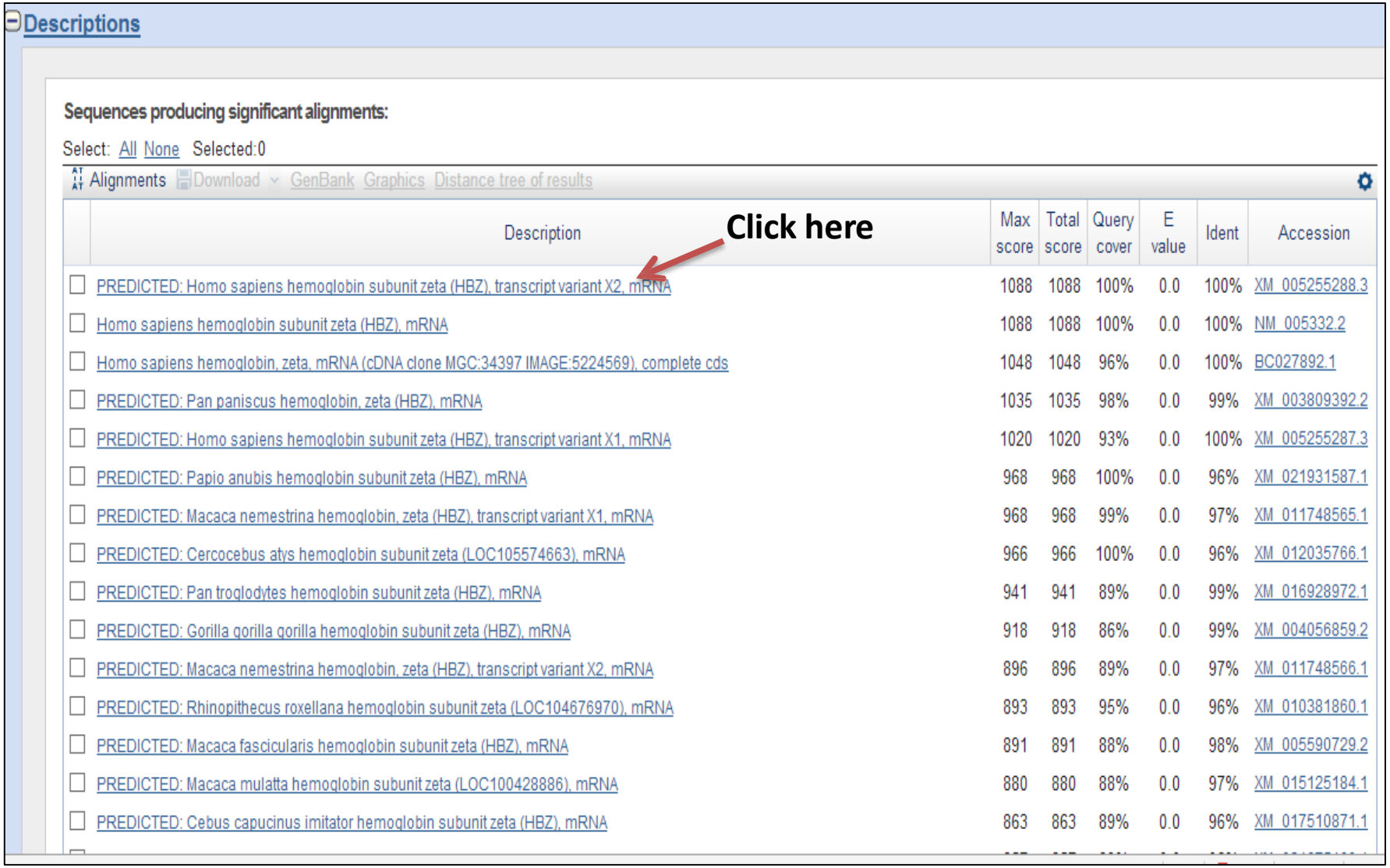
```
>Seq1

ACCAAGGCCAGTCCTGAGCAGGCCCAACTCCAGTGCAGCTGCCCACCCTGCCGCCATGTCTCTGACCAAG ACTGAGAGGACCATCATTGTGTCCATGTGGGCCAAGATCTCCACGCAGGCCGACACCATCGGCACCGAGA
CTCTGGAGAGGCTCTTCCTCAGCCACCCGCAGACCAAGACCTACTTCCCGCACTTCGACCTGCACCCGGG GTCCGCGCAGTTGCGCGCGCACGGCTCCAAGGTGGTGGCCGCCGTGGGCGACGCGGTGAAGAGCATCGAC
GACATCGGCGGCGCCCTGTCCAAGCTGAGCGAGCTGCACGCCTACATCCTGCGCGTGGACCCGGTCAACT TCAAGCTCCTGTCCCACTGCCTGCTGGTCACCCTGGCCGCGCGCTTCCCCGCCGACTTCACGGCCGAGGC
CCACGCCGCCTGGGACAAGTTCCTATCGGTCGTATCCTCTGTCCTGACCGAGAAGTACCGCTGAGCGCCG CCTCCGGGACCCCCAGGACAGGCTGCGGCCCCTCCCCCGTCCTGGAGGTTCCCCAGCCCCACTTACCGCG
TAATGCGCCAATAAACCAATGAACGAAGC
```

- **You will get list of Hits**



- You will see statistic of alignments (Identity, E value)



Description	Max score	Total score	Query cover	E value	Ident	Accession
PREDICTED: Homo sapiens hemoglobin subunit zeta (HBZ), transcript variant X2, mRNA	1088	1088	100%	0.0	100%	XM_005255288.3
Homo sapiens hemoglobin subunit zeta (HBZ), mRNA	1088	1088	100%	0.0	100%	NM_005332.2
Homo sapiens hemoglobin, zeta, mRNA (cDNA clone MGC:34397 IMAGE:5224569), complete cds	1048	1048	96%	0.0	100%	BC027892.1
PREDICTED: Pan paniscus hemoglobin, zeta (HBZ), mRNA	1035	1035	98%	0.0	99%	XM_003809392.2
PREDICTED: Homo sapiens hemoglobin subunit zeta (HBZ), transcript variant X1, mRNA	1020	1020	93%	0.0	100%	XM_005255287.3
PREDICTED: Papio anubis hemoglobin subunit zeta (HBZ), mRNA	968	968	100%	0.0	96%	XM_021931587.1
PREDICTED: Macaca nemestrina hemoglobin, zeta (HBZ), transcript variant X1, mRNA	968	968	99%	0.0	97%	XM_011748565.1
PREDICTED: Cercocebus atys hemoglobin subunit zeta (LOC105574663), mRNA	966	966	100%	0.0	96%	XM_012035766.1
PREDICTED: Pan troglodytes hemoglobin subunit zeta (HBZ), mRNA	941	941	89%	0.0	99%	XM_016928972.1
PREDICTED: Gorilla gorilla gorilla hemoglobin subunit zeta (HBZ), mRNA	918	918	86%	0.0	99%	XM_004056859.2
PREDICTED: Macaca nemestrina hemoglobin, zeta (HBZ), transcript variant X2, mRNA	896	896	89%	0.0	97%	XM_011748566.1
PREDICTED: Rhinopithecus roxellana hemoglobin subunit zeta (LOC104676970), mRNA	893	893	95%	0.0	96%	XM_010381860.1
PREDICTED: Macaca fascicularis hemoglobin subunit zeta (HBZ), mRNA	891	891	88%	0.0	98%	XM_005590729.2
PREDICTED: Macaca mulatta hemoglobin subunit zeta (LOC100428886), mRNA	880	880	88%	0.0	97%	XM_015125184.1
PREDICTED: Cebus capucinus imitator hemoglobin subunit zeta (HBZ), mRNA	863	863	89%	0.0	96%	XM_017510871.1

How well alignment is ? : Bad, Good, Very Good?

PREDICTED: Homo sapiens hemoglobin subunit zeta (HBZ), transcript variant X2, mRNA

Sequence ID: XM_005255288.3 **Length:** 1342 **Number of Matches:** 1

Range 1: 748 to 1336 GenBank Graphics ▼ Next Match ▲ Previous Match

Score	Expect	Identities	Gaps	Strand
1088 bits(589)	0.0	589/589(100%)	0/589(0%)	Plus/Plus

```
Query  1     ACCAAGGCCAGTCCTGAGCAGGCCCAACTCCAGTGCAGCTGCCCACCCTGCCGCCATGTC  60
             ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Sbjct  748   ACCAAGGCCAGTCCTGAGCAGGCCCAACTCCAGTGCAGCTGCCCACCCTGCCGCCATGTC  807

Query  61    TCTGACCAAGACTGAGAGGACCATCATTGTGTCCATGTGGGCCAAGATCTCCACGCAGGC  120
             ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Sbjct  808   TCTGACCAAGACTGAGAGGACCATCATTGTGTCCATGTGGGCCAAGATCTCCACGCAGGC  867

Query  121   CGACACCATCGGCACCGAGACTCTGGAGAGGCTCTTCCTCAGCCACCCGCAGACCAAGAC  180
             ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Sbjct  868   CGACACCATCGGCACCGAGACTCTGGAGAGGCTCTTCCTCAGCCACCCGCAGACCAAGAC  927

Query  181   CTACTTCCCGCACTTCGACCTGCACCCGGGGTCCGCGCAGTTGCGCGCGCACGGCTCCAA  240
             ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Sbjct  928   CTACTTCCCGCACTTCGACCTGCACCCGGGGTCCGCGCAGTTGCGCGCGCACGGCTCCAA  987

Query  241   GGTGGTGGCCGCCGTGGGCGACGCGGTGAAGAGCATCGACGACATCGGCGGCGCCCTGTC  300
             ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Sbjct  988   GGTGGTGGCCGCCGTGGGCGACGCGGTGAAGAGCATCGACGACATCGGCGGCGCCCTGTC  1047

Query  301   CAAGCTGAGCGAGCTGCACGCCTACATCCTGCGCGTGGACCCGGTCAACTTCAAGCTCCT  360
             ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Sbjct  1048  CAAGCTGAGCGAGCTGCACGCCTACATCCTGCGCGTGGACCCGGTCAACTTCAAGCTCCT  1107

Query  361   GTCCCACTGCCTGCTGGTCACCCTGGCCGCGCGCTTCCCCGCCGACTTCACGGCCGAGGC  420
             ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Sbjct  1108  GTCCCACTGCCTGCTGGTCACCCTGGCCGCGCGCTTCCCCGCCGACTTCACGGCCGAGGC  1167

Query  421   CCACGCCGCCTGGGACAAGTTCCTATCGGTCGTATCCTCTGTCCTGACCGAGAAGTACCG  480
             ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Sbjct  1168  CCACGCCGCCTGGGACAAGTTCCTATCGGTCGTATCCTCTGTCCTGACCGAGAAGTACCG  1227

Query  481   CTGAGCGCCGCCTCCGGGACCCCCAGGACAGGCTGCGGCCCCTCCCCCGTCCTGGAGGTT  540
             ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Sbjct  1228  CTGAGCGCCGCCTCCGGGACCCCCAGGACAGGCTGCGGCCCCTCCCCCGTCCTGGAGGTT  1287

Query  541   CCCCAGCCCCACTTACCGCGTAATGCGCCAATAAACCAATGAACGAAGC  589
             ||||||||||||||||||||||||||||||||||||||||||||||||
Sbjct  1288  CCCCAGCCCCACTTACCGCGTAATGCGCCAATAAACCAATGAACGAAGC  1336
```

Resources

- Online Tutorial on Sequence Alignment
  - http://a-little-book-of-r-for-bioinformatics.readthedocs.org/en/latest/src/chapter4.html
- Pairwise alignment of DNA and proteins using your rules:
  - http://www.bioinformatics.org/sms2/pairwise_align_dna.html
- Documentation on libraries
  - Biostings: http://www.bioconductor.org/packages/2.10/bioc/manuals/Biostrings/man/Biostrings.pdf
  - SeqinR: http://seqinr.r-forge.r-project.org/seqinr_2_0-7.pdf