Database Similarity Searches

2020/10/21

1

AN APPLICATION OF PAIRWISE SEQUENCE ALIGNMENT:

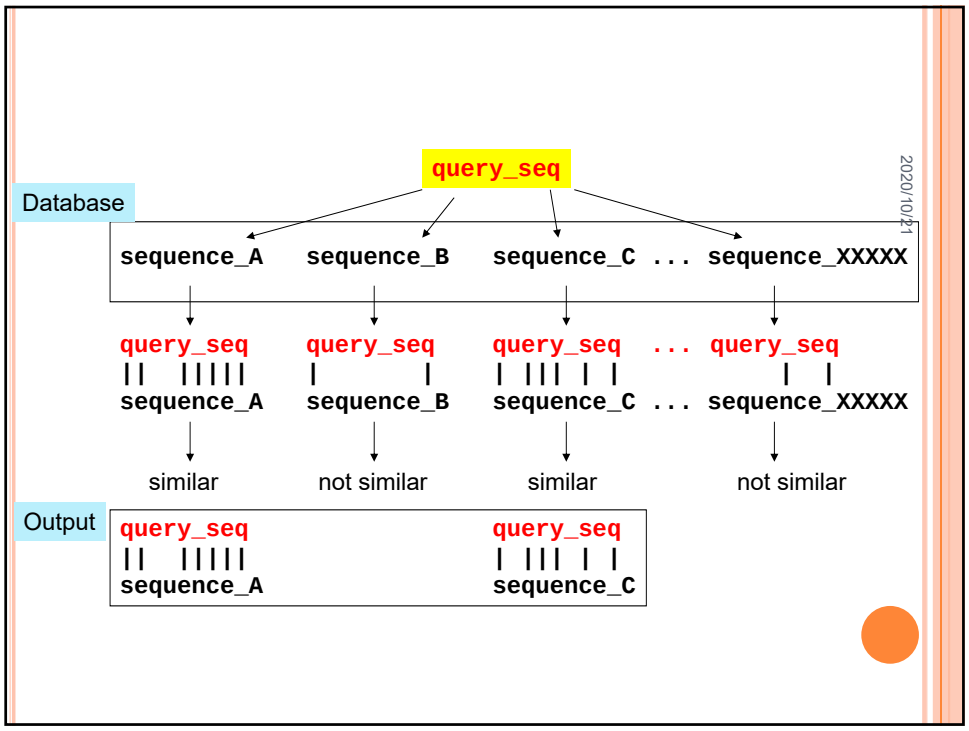
2020/10/21

Retrieving sequences from biological databases based on similarity to a sequence of interest.

The basic procedure for doing this:

- Submit the sequence of interest (the"query" sequence)
- It is aligned in a pairwise manner to EVERY sequence in the database
- Based on these pairwise comparisons, all sequences that have similarity to the query are found
- Pairwise alignments between the query and each of these similar sequences are returned as output

2



Why is it useful to compare a particular sequence to a database of sequences?

- Determine potential function of the query sequence
- Determine homologs of the query (evolutionarily related sequences)

Unique requirements of database searching:

1. sensitivity
2. specificity (selectivity)
3. speed

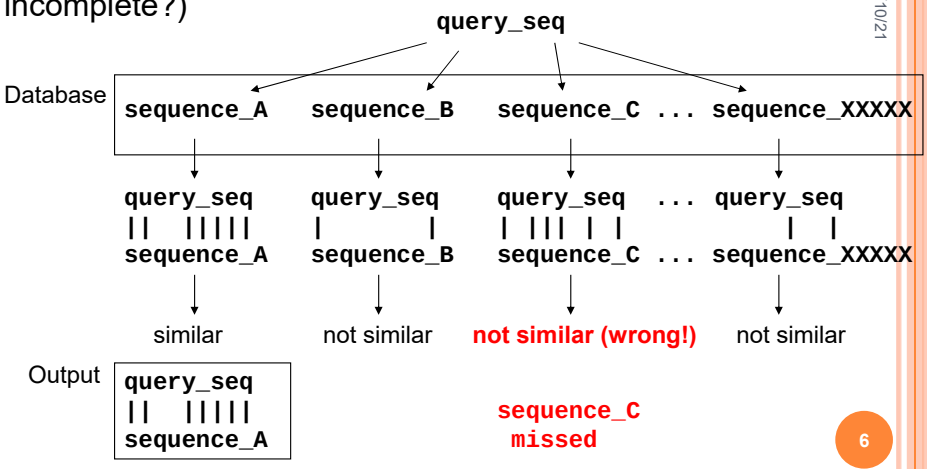
2020/10/21

5

Unique requirements of database searching:

1. sensitivity– how good a method is at identifying sequences from the database that ARE actually similar to the query
(Did it miss some of the similar sequences– is the output incomplete?)

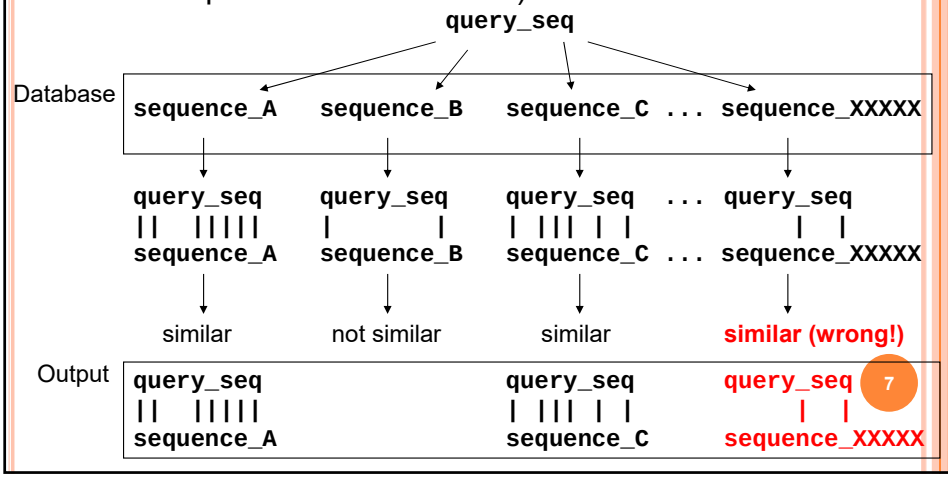
2020/10/21



6

Unique requirements of database searching:

2. specificity– how good a method is at identifying sequences from the database that are NOT actually similar to the query and excluding them from the output (Did it incorrectly determine that some UNRELATED sequences were similar to the query– does the output have "extra stuff"?)



Unique requirements of database searching:

1. sensitivity
2. specificity
3. speed– the time it takes to get the output from the database search (This is an important issue, given the size of sequence databases)

It is difficult to satisfy all three requirements (increasing one tends to decrease another); must compromise to achieve reasonable balance

Algorithms for generating pairwise alignments may use one of three methods:
(1) dot matrix method, (2) dynamic programming method, (3) word method

2020/10/2

Dynamic programming is impractical for pairwise alignment during database searching because it is too SLOW:
- It is computationally intensive (ALL possible alignments are scored)
- A pairwise alignment must be done for EVERY sequence in the database

An estimate made ~10 years ago:
Suppose you have a protein query of 100 residues;
you search a database of 300,000 sequences;
the search would take 2-3 hours using dynamic programming.
Need a method with greater speed, that doesn't sacrifice sensitivity and specificity…

9

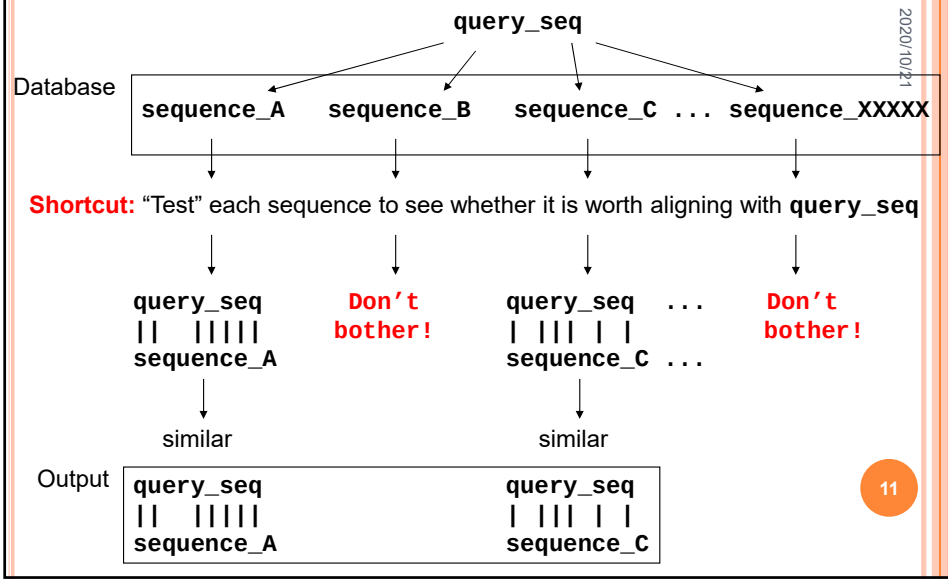
Dynamic programming is an EXHAUSTIVE algorithm:
- Exhaustive algorithms find the best/exact solution to a problem
- Dynamic programming scores ALL possible alignments to find the BEST score

2020/10/21

To solve the speed problem, use a HEURISTIC algorithm:
- Heuristic algorithms find an approximation of the best solution to a problem without exhaustively considering every possible outcome– they do this by taking shortcuts
- Heuristic algorithms are not guaranteed to find the best or most accurate solution

10

A heuristic algorithm will only score SOME alignments, rather than all of them, which saves time; it "takes a shortcut" in order to determine which alignments are worth scoring.



2020/10/21

11

Two commonly used heuristic algorithms for database searching: BLAST and FASTA

- They are not guaranteed to find optimal alignment or to find sequences that are true homologs
- BUT they are 50-100 times faster than dynamic programming
- They have only a moderate decrease in sensitivity and specificity
- They use a heuristic word method

2020/10/21

The heuristic WORD METHOD:

Based on finding short stretches of identical or nearly identical letters (which are called "words", "*k-tuples*", or "*ktups*") in two sequences.

Example:

```
seq1  C U R R E N T L Y T R O P I C A L
seq2  C U R R E N T T O P I C S
```

Both sequences contain the 3-letter "words":

CUR URR RRE ENT OPI PIC

12

The Basics of the Word Method (local alignment method):

- Find word(s) that two sequences have in common and align them
- Extend the alignment by aligning regions of similarity on either side of each word
- Calculate scores of the aligned regions
- Join adjacent high-scoring regions to obtain the final local alignment

2020/10/21

```
query_seq      RMDPYNKLIS   both contain the word "PYN"
database_seq   MHPYNEDIW
                    |
                    ↓
query_seq      RMDPYNKLIS
                  |||       align the word "PYN"
database_seq    MHPYNEDIW
                    |
                    ↓
query_seq      RMDPYNKLIS
               | |||  |    extend the alignment; calculate score
database_seq   MHPYNEDIW    using a specific scoring matrix
```

13

BLAST: Basic Local Alignment Search Tool

- Developed by Stephen Altschul at NCBI in 1990
- One of the most popular and widely used bioinformatics tools because it can accurately detect similarities between DNA or protein sequences quickly without sacrificing sensitivity
- Uses a heuristic word method to align a query sequence with all sequences in a database (local alignment)
- Two versions (which differ slightly):
  - NCBI-BLAST, developed at NCBI:
    - More widely used;
    - Available online at: http://blast.ncbi.nlm.nih.gov/Blast.cgi
  - WU-BLAST, developed at Washington University:
    - Developed from the original NCBI version;
    - Available online at EMBL-EBI: http://www.ebi.ac.uk/Tools/blast2/

2020/10/21

14

How does BLAST work?

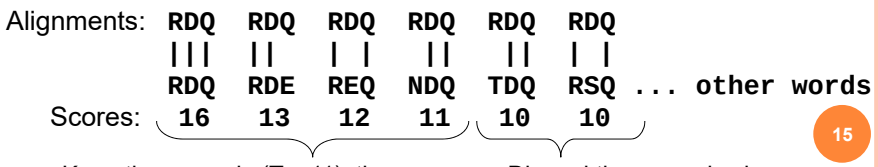
1. Create a list of ALL words found in the query sequence. Usually word size (w) is 3 residues for proteins, 11 for DNA. These words, rather than the full sequence, will be used for searching a sequence database.

Example query sequence: **...RLRDQHK...**
Query words will be (w=3): **RLR, LRD, RDQ, DQH, QHK....**

2020/10/21

2. Determine which words will be "matches" for each query word. There are a total of 8000 possible 3-letter words (20 x 20 x 20). For each query word, create a table of the words (out of the 8000) that align with the query word to give a score ≥ a certain threshold (T); usually T = 11 for proteins. (Choose a scoring matrix to calculate alignment scores; default is BLOSUM62.)

Example: use the query word **RDQ**
Align **RDQ** with each of the 8000 words and calculate scores:

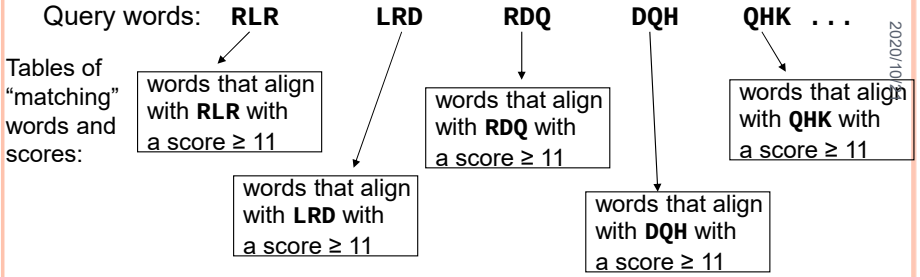


15

Keep these words (T ≥ 11); they will be "matches" to the query word.

Discard these words since score is below T; they are not "matches."

Repeat step 2 for each word in the query sequence. Each query word will then have an associated table of "matching" words with scores above the threshold, T.



2020/10/21

3. Search each sequence in the database for an exact match to any of the query words OR associated matching words (any database sequences that don't contain any of the words can be discarded from further consideration).

4. When a match is found to a database sequence, the word is used to "seed" a possible ungapped alignment between the query sequence and database sequence.

16

Example of seeding:
REQ is similar to the word **RDQ** in the query sequence (T ≥ 11).
REQ occurs in a particular database sequence:
Align these words:

```
Query_sequence: TDKRPFIETAERLRDQHKKDYPEYKYQPRRRKNGK
       Matches:                R+QHKKD+P+YKYQPRRRK
  Database_seq: GEKRPFVEGAERLREQHKKDHPDYKYQPRRRKSVK
```

join

2020/10/21

If another word match is found "near" this one, the two are joined to form a longer ungapped region of alignment. (Earlier versions of BLAST did not require two proximal words in a database sequence.)

5. The aligned region is extended on either side until the total alignment score begins to drop due to mismatches. This aligned region is called a high-scoring segment pair (HSP).

```
Query_sequence: TDKRPFIETAERLRDQHKKDYPEYKYQPRRRKNGK
       Matches:  +KRPF+E AERLR+QHKKD+P+YKYQPRRRK+ K
  Database_seq: GEKRPFVEGAERLREQHKKDHPDYKYQPRRRKSVK
```

extend extend

HSP

17

6. If the HSP's alignment score is greater than a certain cutoff value (S), it is kept. If the HSP's score is less than this cutoff, this alignment is discarded.

```
Query_sequence: TDKRPFIETAERLRDQHKKDYPEYKYQPRRRKNGK
       Matches:  +KRPF+E AERLR+QHKKD+P+YKYQPRRRK+ K
  Database_seq: GEKRPFVEGAERLREQHKKDHPDYKYQPRRRKSVK
```

HSP: Is score ≥ S?
If yes, keep.
If no, discard.

2020/10/21

7. If multiple HSP's are found for a single database sequence, they may be connected to generate a longer, gapped alignment. Thus, BLAST produces gapped local alignments. (Earlier versions of BLAST did not have this step.)

8. A Smith-Waterman local alignment is generated for the query sequence and each matching database sequence. The BLAST output shows these alignments. The matching database sequences are called "hits" to the query sequence. They are the sequences that are similar (and possibly homologous) to the query.

18

Effect of Changing Threshold Values on a Protein BLAST Search

	T = 11 (default)	T = 5	T = 17
# of sequences in database	1,046,476	1,046,476	1,046,476
# of hits to database	129,839,417	2,200,945,350	12,002,487
# of extensions	5,198,652	589,935,555	61,838
# of successful extensions	8,377	13,145	1,117
# of HSP's gapped	145	146	93

2020/2/21

Think about the trend in # of hits to database . . . we will discuss this in class.

Note that the final results are similar for default threshold and lower threshold of 5 (although T= 5 will be slower).

But some hits are missed with higher threshold of 17.

19

Table 4-3 from *Bioinformatics and Functional Genomics*, by J. Pevsner

But how do we know if an alignment obtained from BLAST is statistically significant? Can we infer that the two sequences are homologous?

2020/10/21

Maybe two unrelated sequences could be aligned with a score that is just as good…

```
Query_sequence: TDKRPFIETAERLRDQHKKDYPEYKYQPRRRKNGK
       Matches:  +KRPF+E AERLR+QHKKD+P+YKYQPRRRK+ K
Database_seq: GEKRPFVEGAERLREQHKKDHPDYKYQPRRRKSVK
```

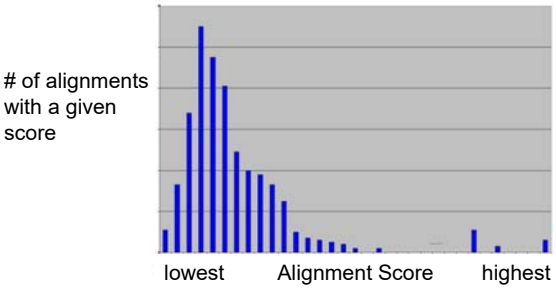
HSP: score = S

Calculate a **P-value** to determine the chances that a score ≥ S could be obtained by aligning two unrelated sequences.

If the chances are low (small P-value), then it is "safe" to conclude that this alignment is not due to chance and to infer that the sequences are homologous.

20

For the comparison of a query sequence to a database of random sequences of uniform length, the alignment scores follow the **Extreme Value Distribution:**



2020/10/21

The **P-value** of a given alignment score indicates the **probability that the alignment is due to chance** (the smaller the P-value, the less likely the alignment is due to chance).

$$P(x \geq S) = 1 - \exp[-Kmne^{-\lambda S}]$$

m and *n* are the lengths of the two sequences being compared;
K and λ are constants that depend on the scoring matrix used.

21

In the context of a database search the P-value is:

$$P(x \geq S) = 1 - \exp[-Km'n'e^{-\lambda S}]$$

2020/10/21

n' is the "effective" length of the query sequence (actual length, *n*, minus average length of an alignment between two random sequences of lengths *m* and *n*)

m' is the "effective" length of entire database (in residues) (actual length, *m*, minus average length of an alignment between two random sequences of lengths *m* and *n*)

K and λ are constants that depend on scoring matrix used

22

BLAST Search Statistics

The following tables are part of the output from a real BLAST search. Note the items that are marked.

2020/10/21

Search Parameters	
Program	blastp
Word size	3
Expect value	10
Hitlist size	100
Gapcosts	11, 1 (opening, extension)
Matrix	BLOSUM62
Threshold	11
Composition-based stats	2
Filter string	F
Genetic Code	1
Window Size	40

T

23

See Fig. 4.16 in *Bioinformatics and Functional Genomics*, by J. Pevsner for more info.

Database	
Posted date	Jun 29, 2009 5:41 PM
Number of letters in database	3,163,461,954
Number of sequences in database	9,230,955
Entrez query	none

2020/10/21

Karlin-Altschul statistics		
Params	**Ungapped**	**Gapped**
Lambda	0.319407	0.267
K	0.137232	0.041
H	0.427596	0.14

λ

Results Statistics	
Length adjustment	142
Effective length of query	414
Effective length of database	1852666344
Effective search space	767003866416
Effective search space used	767003866416

n'

m'

24

The SIZE of the DATABASE is important:
The larger the database being searched, the more unrelated sequences there are, and the greater the chances that you will find a high-scoring match between the query and an unrelated sequence.

2020/10/21

For a given database size, we can calculate the number of high-scoring matches we expect to observe between the query and unrelated sequences. That number is called the **E-value** (**expected value**).

E = $Km'n'e^{-\lambda S}$ or **E = *pD*** (when $p < 0.1$)

D = number of sequences in the database (database "size")
p = probability, according to Extreme Value Distribution, of obtaining an alignment score ≥ S by chance
m' = effective length of the database in residues (another measure of database size)

25

A p-value is a way of representing the significance of an alignment. An E-value is simply another way of representing the significance of that alignment.

2020/10/21

HOW TO INTERPRET AN E-VALUE:

The **E-value** is **the number of HSP's** with an alignment score ≥ S that are expected to occur by chance when searching a database of *D* sequences.

Suppose we found an HSP with score = 20 when searching a certain database:

- If E = 4, then we expect to find four HSP's by chance that have scores ≥ 20 when searching this database using the same parameters.
- If E = 0.01, then we expect to find 0.01 HSP's by chance that have scores ≥ 20 when searching this database using the same parameters.

26

E-values are shown on BLAST output using a rather strange notation. For example, an E-value may be shown as: 1e-5

The notation "1e-5" means 1×10^{-5}

2020/10/21

Interpreting E-values in Terms of Sequence Homology

E-values	Interpretation
E-value < 1e-50	high confidence the match is NOT due to chance ("safe" to infer homology)
E-value 0.01 to 1e-50	"safe" to infer homology
E-value 10 to 0.01	match is not significant, but possible distant homologs
E-value > 10	sequences may be related by chance

(the smaller the E-value, the less likely the alignment is due to chance)

27

E-values are useful for evaluating the significance of an alignment resulting from a database search.

HOWEVER, E-values obtained when searching one database cannot be compared to those obtained when searching another database or when using a different scoring matrix and gap penalties.

2020/10/21

↓

BECAUSE the E-value will increase as the size of the database increases.

E = *pD*

28

Is it valid to compare the scores of two alignments from different BLAST searches to determine which alignment is "better"?

- The raw score of an alignment depends on the scoring matrix and gap penalties used.
- BLAST output shows normalized scores in addition to raw scores; these normalized scores are independent of the scoring matrix.

2020/10/21

Calculation of a normalized score (*S*'):

$$S' = (\lambda S - \ln K) / \ln 2$$

S = raw alignment score
λ = constant that depends on scoring matrix
K = constant that depends on scoring matrix

S' accounts for the scoring system that was used because it incorporates λ and *K*.

29

Example:

```
 query_sequence  Y C D A
        matches  +   +
  database_seq  F M E G
BLOSUM62 scores: 3-1 2 0
```

2020/10/21

(Scores in BLOSUM matrices are in "half bits.")
("bits" = logarithms to the base 2)
(Scores in PAM matrices are base 10)

Raw score: 3 – 1 + 2 + 0 = 4 half bits = **2 bits**

Normalized score: (Using λ = 0.32 and *K* = 0.136)

[(0.32 x **2 bits**) – ln(0.136)] / ln2 = **3.8 bits**

30

Low Complexity Regions (LCRs):

Regions that contain highly repetitive residues, and therefore, have low information content. LCRs may be:

- short repeating segments
- segments that contain an overrepresentation of a small number of residues

LCRs may account for 15% of total protein sequences in public databases.

LCRs in the query sequence can lead to spurious hits and artificially high alignment scores with unrelated sequences.

2020/10/21

31

Two options for filtering/masking LCRs in the query sequence during BLAST:

Filter (Low complexity regions):

- Characters in LCRs are ignored by BLAST and not used in the alignment process
- In the output, LCR regions of the query are replaced with lower-case characters or an ambiguous character (X for proteins, N for DNA)

Mask (Mask for lookup table only):

- Characters in LCRs are ignored in constructing the lookup table of words, but are used in word extension and optimization of alignments

(Of course it is possible that authentic matches may be missed when filtering is applied)

2020/10/21

32

There are more than seven versions of BLAST available at NCBI:

1. BLASTP
2. BLASTN and megablast
3. BLASTX
4. TBLASTN
5. TBLASTX
6. bl2seq (pairwise alignment of two sequences)
7. PSI-BLAST

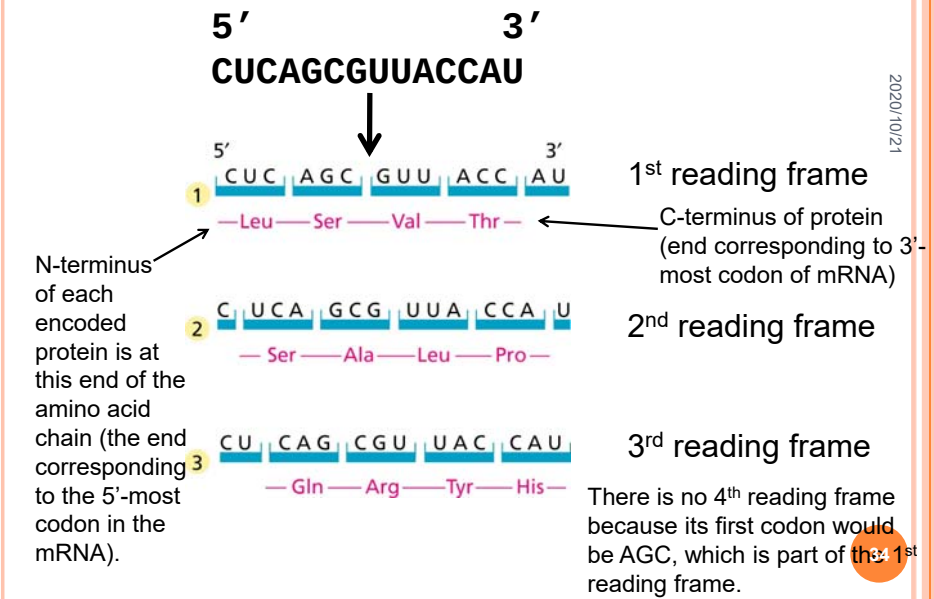
2020/10/21

The first five versions are described on one of the following slides. PSI-BLAST will be described later in the course.

Before the description of the first five versions of BLAST, here is a review of reading frames…

33

Recall that a strand of mRNA has three reading frames:



2020/10/21

34

Fig. 1.9 from *Understanding Bioinformatics* by M. Zvelebil and J. O. Baum

Double-stranded DNA can potentially encode six different proteins (since each strand has 3 reading frames):

2020/10/21

```
5′                                                              3′
ATGAAGTGGGTGTGGGCGCTCTTGCTGTTGGCGGCGTGGGCAGCGGCCGAG
-+---------+---------+---------+---------+---------+---------
TACTTCACCCACACCCGCGAGAACGACAACCGCCGCACCCGTCGCCGGCTC
3′                                                              5′
```

Top strand has 3 reading frames (5' to 3') and each encodes a different potential protein:

1st reading frame: 5′ ATG AAG TGG...etc. 3′ → M K W... etc. (protein1)
2nd reading frame: 5′ TGA AGT GGG...etc. 3′ → * S G... etc. (protein2)
3rd reading frame: 5′ GAA GTG GGT...etc. 3′ → E V G... etc. (protein3)

Bottom strand has 3 reading frames (5' to 3') and each encodes a different potential protein:

1st reading frame: 5′ CTC GGC CGC...etc. 3′ → L G R... etc. (protein4)
2nd reading frame: 5′ TCG GCC GCT...etc. 3′ → S A A... etc. (protein5)
3rd reading frame: 5′ CGG CCG CTG...etc. 3′ → R P L... etc. (protein6)

35

From Fig. 4.4 from *Bioinformatics and Functional Genomics* by J. Pevsner

The six potential proteins encoded by the six reading frames are shown below. Each amino acid is "lined up" directly below the first nucleotide of the codon encoding it. A few are color-coded so you can see how this works. Compare this to the previous slide.

2020/10/21

```
5′                                                              3′
ATGAAGTGGGTGTGGGCGCTCTTGCTGTTGGCGGCGTGGGCAGCGGCCGAG
-+---------+---------+---------+---------+---------+---------
TACTTCACCCACACCCGCGAGAACGACAACCGCCGCACCCGTCGCCGGCTC
3′                                                              5′

N-term                                                  C-term
M  K  W  V  W  A  L  L  L  L  A  A  W  A  A  A  E
 *  S  G  C  G  R  S  C  C  W  R  R  G  Q  R  P
  E  V  G  V  G  A  L  A  V  G  G  V  G  S  G  R
-+---------+---------+---------+---------+---------+---------
  H  L  P  H  P  R  E  Q  Q  Q  R  R  P  C  R  G  L
    F  H  T  H  A  S  K  S  N  A  A  H  A  A  A  S
   S  T  P  T  P  A  R  A  T  P  P  T  P  L  P  R
C-term                                                  N-term
```

Top strand of DNA encodes protein left to right (5' to 3')

Bottom strand of DNA encodes protein right to left (also 5' to 3')

36

From Fig. 4.4 from *Bioinformatics and Functional Genomics* by J. Pevsner

Program	Query	Number of database searches	Database
BLASTP	protein	1	protein
Use blastp to compare a protein query to a database of proteins			
BLASTN	DNA	2	DNA
Use blastn (or megablast) to compare a DNA query against both strands of a DNA database			
BLASTX	DNA	6	protein
Blastx translates a DNA sequence into 6 protein sequences using all 6 possible reading frames, and then compares each of these proteins to a protein database			
TBLASTN	protein	6	DNA
Tblastn translates every DNA sequence in a database into 6 potential proteins, and then compares the protein query against each of those translated proteins			
TBLASTX	DNA	36	DNA
Tlastx is the most computationally intensive blast algorithm. It translates DNA from both a query and a database into 6 potential proteins, then performs 36 protein-protein database searches			

2020/10/21

37

From Fig. 4.3 from *Bioinformatics and Functional Genomics* by J. Pevsner

FASTA: FAST-All

- Developed by David Lipman and William Pearson in 1988
- The first widely used program for rapid database searching
- Uses a heuristic word method to create local alignments of a query sequence with database sequences
- Begins by looking for exact matches of words in the two sequences (*ktup* = 1 or 2 for protein, and 4-6 for DNA)

- Available online at University of Virginia (W. Pearson):
  http://fasta.bioch.virginia.edu/fasta_www2/fasta_list2.shtml
- Available online at EMBL-EBI:
  http://www.ebi.ac.uk/Tools/fasta/index.html

2020/10/21

38

Comparison of BLAST and FASTA:
Several published studies have performed analyses to determine which algorithm performs better in various scenarios.

2020/10/21

- BLAST is faster
- FASTA generally produces a better final alignment
- FASTA is more likely to find distantly related sequences
- Performance of the two methods is similar for highly similar sequences

- Both methods are appropriate for rapid initial searches

39

Database Searching Using an Exhaustive Algorithm:

In practice, BLAST and FASTA are usually successful in finding sequences in a database that are related to a query.

2020/10/21

BUT they are not guaranteed to find all related sequences in a database or to produce the best alignment because they are heuristic methods.

The Smith-Waterman dynamic programming algorithm provides the most reliable method for finding related sequences in a database search.

Parallel computing has made it possible to use the Smith-Waterman algorithm for database searching in a reasonable timeframe (but not for routine use).

40

SSEARCH at the University of Virginia is based on S-W algorithm:
http://fasta.bioch.virginia.edu/fasta_www2/fasta_list2.shtml