CMPT 830 - Bioinformatics and Computational Biology

Chapter 3: Sequence Alignment

Ian McQuillan and Tony Kusalik
mcquillan@cs.usask.ca

Department of Computer Science
University of Saskatchewan

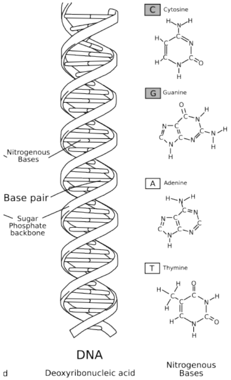
September 24, 2019

DNA

- Holds genetic information passed by inheritance.
- Determines the development and function of living organisms.
- Organized into chromosomes and stored in the nucleus of eukaryotes, and in organelles such as mitochondria.

DNA

- Deoxyribonucleic acid is a macromolecule composed of chains of nucleotides.
- Each nucleotide is composed of a nitrogenous base, a five-carbon sugar, and phosphate group.



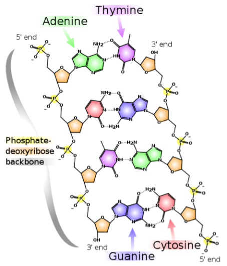
http://en.wikipedia.org/wiki/File:Nucleotides_1.svg

DNA Composition

- A chain of nucleotides has a sugar-phosphate backbone.
- There is directionality to single-stranded DNA, with one end called the $5'$ end, and the other the $3'$ end.
- Usually we write DNA as a sequence (or a word, or string) using the letters A,G,C,T representing the four bases.
- By convention, usually we write a sequence from the $5'$ end to the $3'$ end.
- DNA usually exists as a pair of molecules.

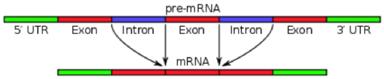
DNA Composition

- Double-stranded DNA contains two single-stranded chains of nucleotides that run in opposite directions.
- The two strands are held together by hydrogen bonds between pairing bases.
- A (adenine) pairs with T (thymine), and C (cytosine) pairs with G (guanine).



Genes

- A sequence of DNA that codes for a protein is called a gene.
- While the human genome contains 3.2 billion base pairs, there are less than $30,000$ genes.
- Over 98% of human DNA does not code for genes.
- Eukaryotic genes can contain introns and exons, with introns getting spliced out before translation into protein.



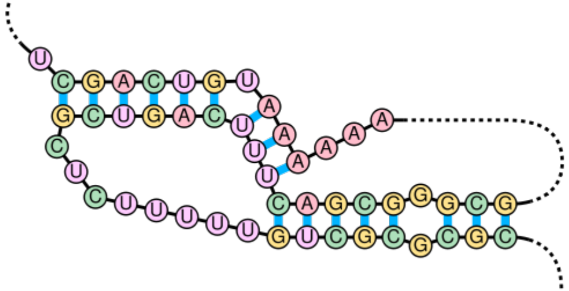
picture from http://en.wikipedia.org/wiki/Intron#mediaviewer/File:Pre-mRNA_to_mRNA.svg

RNA

- RNA is a type of nucleic acid like DNA, but is mainly single-stranded and shorter.
- The base complementary to adenine is uracil instead of thymine.
- Many types exist such as messenger RNA (mRNA), transfer RNA (tRNA), ribosomal RNA (rRNA).

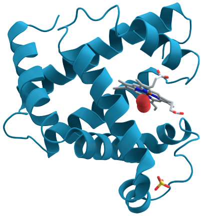
RNA

- RNA is similar to DNA, however, the sugar is slightly different and the base thymine (T) is usually replaced by uracil (U).
- RNA is usually single-stranded.
- The structure of RNA can take on a variety of interesting shapes (more later in the course).



picture from http://en.wikipedia.org/wiki/File:Pseudoknot.svg

Proteins

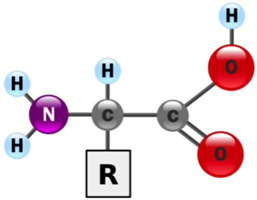


- Proteins are macromulecules consisting of chains of amino acids.
- There are 20 main amino acids.
- The sequence of amino acids in a protein is known as the *primary structure*.
- A protein folds into a complex 3-dimensional shape, the *tertiary structure*.

picture from `http://upload.wikimedia.org/wikipedia/commons/6/60/Myoglobin.png`

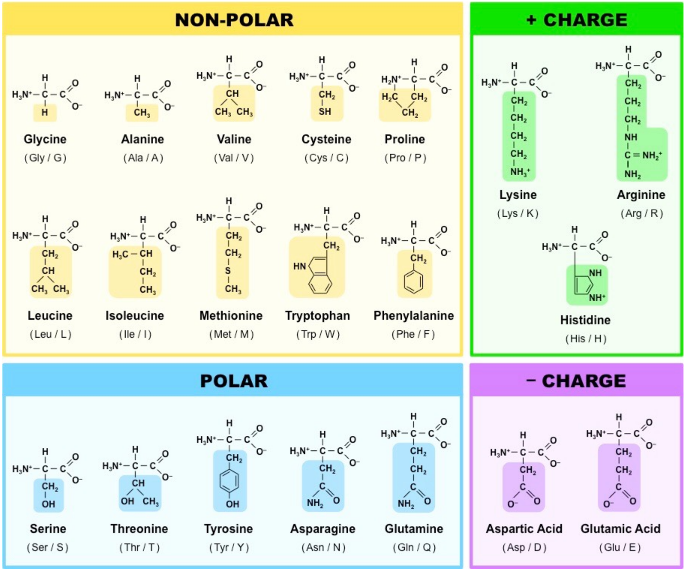
Amino Acids

- Amino acids are molecules that contain an amine (NH2) group, a carboxyl group (COOH) and a side chain (R).
- Functional groups differ between amino acids.



picture from http://upload.wikimedia.org/wikipedia/commons/c/ce/AminoAcidball.svg

Amino Acids



picture from
socratic.org/questions/which-part-of-an-amino-acid-s-structure-makes-it-unique-from-other-amino-acids

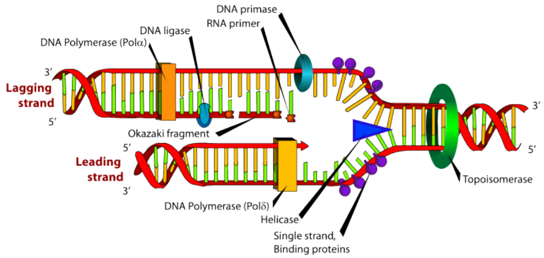
DNA - RNA - Protein

information transfer

- DNA can be copied to DNA through *DNA replication*,
- DNA can be “copied” into RNA through *transcription*,
- RNA can be transformed into proteins through *translation*.

There are also less frequent changes, which change RNA into DNA, RNA into RNA and DNA into protein.

DNA replication



- DNA replication is accurate, however there are occasionally errors created in the process. Sometimes, an incorrect nucleotide will get substituted.
- Organisms have error-correction mechanisms that usually (but not always) catch such errors.
- The ones that get through result in *mutations*.

picture from http://en.wikipedia.org/wiki/File:DNA_replication_en.svg

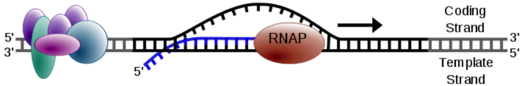
Changes to DNA

- *Mutations* occur when one nucleotide gets substituted for another.
- They are usually the result of a malfunction in the DNA replication, which could not get repaired.
- *Insertions* can add one or more nucleotides into DNA. These are usually caused by, so-called *transposable elements*.
- *Deletions* remove one or more nucleotides.

DNA Transcription

- Transcription synthesizes RNA from a DNA template.
- The process is similar to DNA replication, but with a different set of enzymes.
- RNA polymerase is a key enzyme.

Transcription



- A gene is located on the coding or sense strand of DNA.
- The template strand is called the antisense strand.
- The $5'$ end is upstream and $3'$ is downstream for each DNA strand.

http://upload.wikimedia.org/wikipedia/commons/2/21/Simple_transcription_elongation1.svg

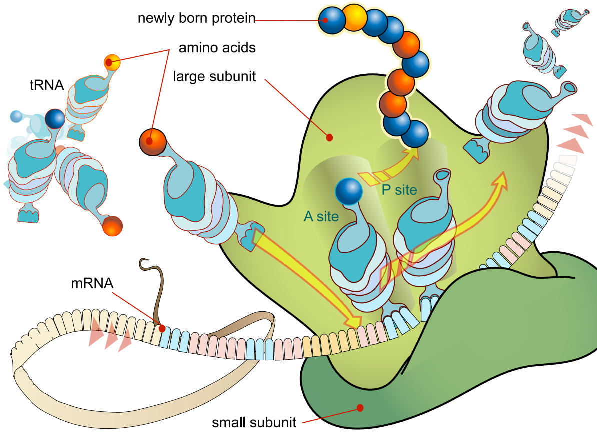
Transcription

- For the most part, DNA gets copied into RNA in a straightforward fashion (algorithmically).
- The base uracil is used instead of thymine, but otherwise, the bases remain unchanged from the DNA.
- Entire chromosomes do not get transcribed, but smaller “segments” of chromosomes do.
- Thus, the tricky part involves the cellular machinery knowing where transcription should start and end.

More on this later.

Translation

RNA is translated into amino acid sequence in a less straightforward way.



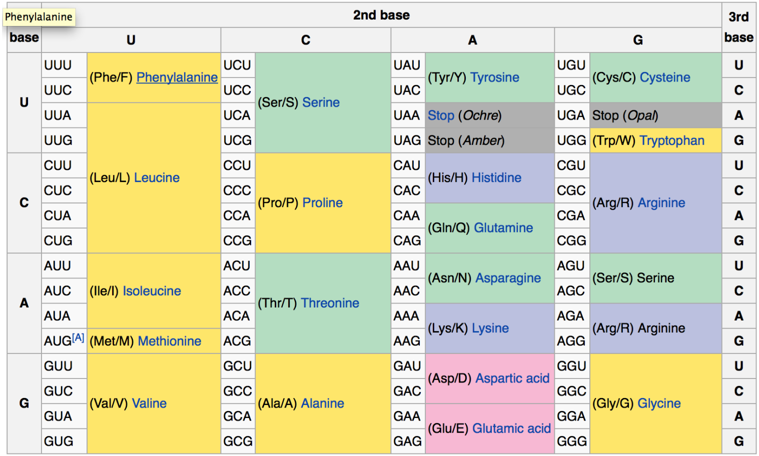
picture from http://en.wikipedia.org/wiki/File:Ribosome_mRNA_translation_en.svg

Translation

- RNA is translated into amino acid sequence in a less straightforward way.
- There are 20 main amino acids, but only four ribonucleotides.
- Thus, 1 nucleotide is not enough to uniquely identify an amino acid, as $4^1 = 4 < 20$.
- Similarly, $4^2 = 16 < 20$.
- However, $4^3 = 64 > 20$.
- Indeed, three consecutive bases, called a codon, uniquely determines an amino acid.

Translation

- Most amino acids have multiple codons which code for them.



Phenylalanine	2nd base								3rd
base	U		C		A		G		base
U	UUU	(Phe/F) Phenylalanine	UCU	(Ser/S) Serine	UAU	(Tyr/Y) Tyrosine	UGU	(Cys/C) Cysteine	U
	UUC		UCC		UAC		UGC		C
	UUA	(Leu/L) Leucine	UCA		UAA	Stop (*Ochre*)	UGA	Stop (*Opal*)	A
	UUG		UCG		UAG	Stop (*Amber*)	UGG	(Trp/W) Tryptophan	G
C	CUU		CCU	(Pro/P) Proline	CAU	(His/H) Histidine	CGU	(Arg/R) Arginine	U
	CUC		CCC		CAC		CGC		C
	CUA		CCA		CAA	(Gln/Q) Glutamine	CGA		A
	CUG		CCG		CAG		CGG		G
A	AUU	(Ile/I) Isoleucine	ACU	(Thr/T) Threonine	AAU	(Asn/N) Asparagine	AGU	(Ser/S) Serine	U
	AUC		ACC		AAC		AGC		C
	AUA		ACA		AAA	(Lys/K) Lysine	AGA	(Arg/R) Arginine	A
	AUG[A]	(Met/M) Methionine	ACG		AAG		AGG		G
G	GUU	(Val/V) Valine	GCU	(Ala/A) Alanine	GAU	(Asp/D) Aspartic acid	GGU	(Gly/G) Glycine	U
	GUC		GCC		GAC		GGC		C
	GUA		GCA		GAA	(Glu/E) Glutamic acid	GGA		A
	GUG		GCG		GAG		GGG		G

picture from http://en.wikipedia.org/wiki/Genetic_code

Proteins

- We've seen that there are 20 main amino acids.
- Proteins fold into specific three-dimensional structures, which largely impacts their function.
- They do almost all essential work for the cell including building cellular stuctures, performing metabolic functions and mediating information flow.

Lots more on this later.

Guess a word?

Let's play a game:

- We all write down 8 arbitrary letters (as a "word") on a piece of paper.
- Either make up a string randomly, or copy off someone else.
- That's the game.

How did you make them up?

- You could just think of letters (sort of randomly) and jot them down.
- Or you could see what your neighbour is writing and copy that. In this case, there is a evolutionary relatedness to the words if you copy. One depends on the other.

Check for copying

- How do we check for copying?
- How many 8 letter words are there?

Check for copying

- There are $26^8 \approx 208$ billion words[9]. If they are all random, there's a 1 divided by 208 billion chance of picking any given random word.
- The chances of two given people randomly guessing the same word is $\frac{1}{208{,}827{,}064{,}580}$.
- About as likely as winning the lottery where you have 1 ticket and everyone else in on earth has 30 tickets.

[9] 26 possible English letters at 8 positions

Check for Copying

- But what if my word is *kwjofjsa* and someone else's is *knjofjsa*?
- Maybe someone copied me but accidentally changed a letter, or has imperfect eyesight?
- There are 8 positions where there could be one "mutation", and 25 possible mutations at that position.
- So there are $25 \cdot 8 = 200$ words that are off from my word by 1 mutation.
- Odds you picked one is $\frac{200}{208{,}827{,}064{,}580} \approx \frac{1}{1{,}044{,}135{,}323}$.
- It seems someone copied.

Check for Copying

- But what if my word is *kwjofjsa* and someone else's is *knjofjwa*?
- For mutations at 2 positions, there are $\binom{8}{2} \cdot 25 \cdot 25 = \frac{(8 \cdot 7 \cdot 25 \cdot 25)}{2}$ possible words off by 2 mutations.
- So, chances are $\frac{17{,}500}{208{,}827{,}064{,}580} \approx \frac{1}{11{,}932{,}975}$ that someone randomly picked one.
- It seems someone copied rather than the unlikely possibility of coming up with such a close "word" at random

Check for Copying

- Mutations at 3 positions, 5 positions same, still an excellent $\frac{1}{238{,}659}$.
- Mutations at 4 positions, 4 positions same, still a pretty good $\frac{1}{7637}$.
- Mutations at 5 positions, 3 positions same, about $\frac{1}{382}$.
- Mutations at 6 positions, 2 positions same, about $\frac{1}{31}$.
- With 2 letters in common, likely someone copied.
- Having only 3 letters in common, quite likely someone copied.

Similarity $\rightarrow$ Relatedness

- The key here is that similarity can provide evidence that words have some “relatedness”.
- Amount of relatedness depends on the amount of similarity. The more similar the words are, the stronger evidence there is that they are related.
- As seen above, in the case of having just a few letters in common between two sequences, it is more probably that the two sequences are related (“someone copied”) than it is that it occurred by random chance

Similarity $\rightarrow$ Relatedness

- But similarity and relatedness are two different things.
- One could quantify how similar two words are (by perhaps counting mutations).
- But they are either related or not (they copied or they didn't).
- Even if two words are very similar, there could have been copying that occurred, or the two people could have, by chance, guessed similar words.

Some Terminology

Definition: Homology

Homology is the existence of a relationship between two or more genes or proteins that can be attributed to descent from a common ancestor.

- This is the concept of “relatedness” that we have been using.
- There is homology if there is an evolution from a common ancestor to the sequences.
- In the word game, there is a common ancestor (the first person who wrote the word) that led to the homology between all the words derived from the first word.

Homology

Homologs

Two (or more) sequences are *homologs* if there is homology between them.

Example

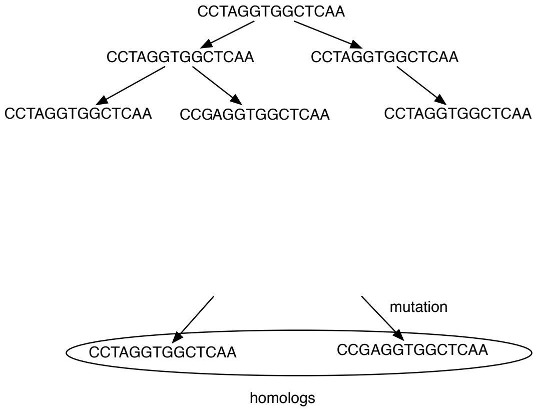
- At one point in time, there is a gene with this fragment: `CCTAGGTGGCTCAA`.
- Then, this gene duplicates a million times through DNA replication.
- Then after one million copies have been created, one instance of this gene has an error, mutating the first `T` to a `G`.

Homologs

Example

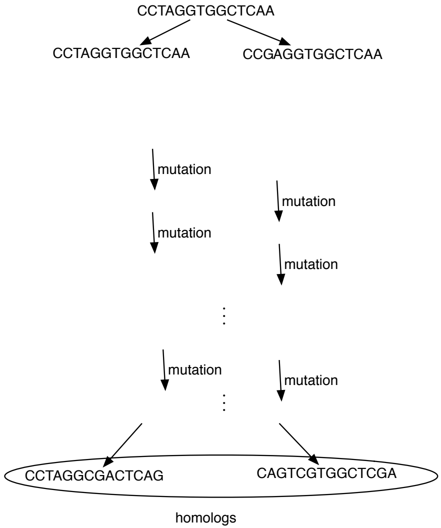
- Then there are many copies of `CCTAGGTGGCTCAA`.
- And one copy of `CCGAGGTGGCTCAA`.
- The gene with the first and the gene with the second are homologs. They descended from a shared ancestor.
- In this case, the shared ancestor is the same sequence as one of the homologs.

Homologs



In this instance, the shared ancestor is the same sequence as one of the current sequences.

Homologs



In this instance, both homologs are different from the shared ancestor.

Types of Homologs

Orthologs

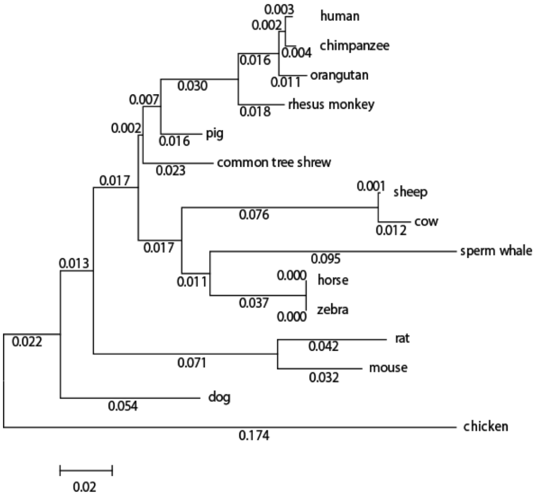
Orthologs are homologs in different species that arose from a common ancestral gene during speciation. Orthologs may or may not be responsible for a similar function.

Paralogs

Paralogs are homologs within a single species that arose by gene duplication.

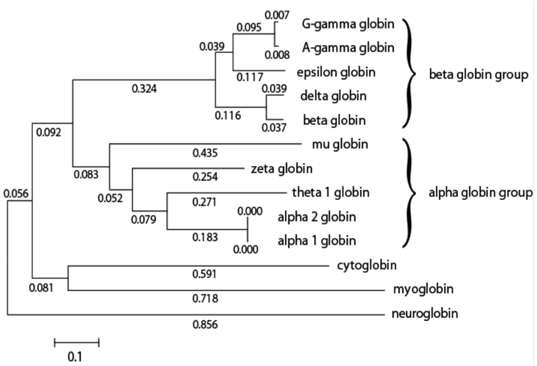
With paralogs, there will be homologs at separate loci of a genome.

Orthologs



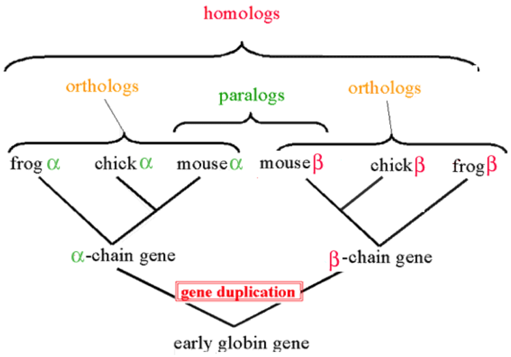
Orthologs: members of a gene (protein) family in various organisms.
This tree shows globin orthologs.

Paralogs



Paralogs: members of a gene (protein) family within a species. This tree shows human globin paralogs.

Both



Orthologs and paralogs are often viewed in a single tree.

image from http://www.bio.davidson.edu/courses/molbio/molstudents/spring2010/rydberg/orthologs3.jpg

Homology Sometimes Implies Function

- If two genes are similar, they are often homologous.
- If two genes are homologous, they often (but not always) have a similar function.

Dot Plots

- An easy way to **visualize** similarity between two sequences is through the use of *dot plots*.
- With two sequences, we make a 2D plot with one sequence along the y-axis and a second sequence along the x-axis.
- If row i of the plot has the same character as column j, then we place a dot at position (i, j), and leave blank otherwise.

Dot Plots

Example

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

a dot is placed in a position if the indexing row and column are labelled with the same letter.

Dot Plots

- If the sequences were identical, then the diagonal would be completely filled.
- Since we are looking for “similarity”, we are looking for “almost” a diagonal.
- Or, at least a path from the top-left corner to the bottom-right corner that is “closest” to a solid diagonal.
- These dot-plots can get quite noisy, since any given nucleotide from one sequence will appear all over the other sequence (at random, every four nucleotides).

Dot Plots

So, typically, we do not just compare single nucleotides, but we use a "sliding window" and a "threshold".

Example

- Assume the window size is 10 and the threshold is 8.
- Start by comparing nucleotides $1 - 10$ of the x-axis with $1 - 10$ of the y-axis.
- If they have at least 8 nucleotides in common, then we put a dot in position $(1, 1)$.
- Then we compare nucleotides $1 - 10$ on the x-axis with $2 - 11$ from the y-axis and put the result in $(1, 2)$, and so on.

By adjusting window size and threshold, much of the noise can be eliminated and the similarity between sequences is clarified.

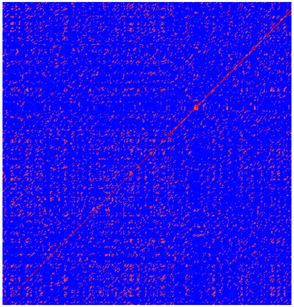
Dot Plots

- It's fun to play with dot plots!
- EMBOSS suite: http://emboss.sourceforge.net/apps/release/6.5/emboss/apps/alignment_dot_plots_group.html
- Visual interface to EMBOSS program: http://emboss.bioinformatics.nl/cgi-bin/emboss/dotmatcher
- dotlet: http://emboss.bioinformatics.nl/cgi-bin/emboss/dotmatcher (in Java) or https://dotlet.vital-it.chr (in JavaScript)

Dot Plots

- Several illustrative examples at http://resources.qiagenbioinformatics.com/manuals/clcgenomicsworkbench/650/Examples_interpretations_dot_plots.html

- E.g. using sequences of two genes representing two Infuenza A virus nucleoproteins infecting ducks and chickens.



Similarity Through Sequence Alignment

- The most common way of assessing similarity is through a *sequence alignment*.
- A sequence alignment is a great tool to understand how similar two sequences are to each other, and to see the parts that are similar and the parts that are not similar.
- It is likely the most common task performed in bioinformatics.

Pairwise Sequence Comparison

- Sequence comparison can help with determining the
    1. function of a new sequence,
    2. evolutionary relationships (genes, proteins, species),
    3. predicting structure and function of proteins.
- We've already seen how similarity can be used to provide evidence of evolutionary relationships, or homology (with the word game).

Homology Sometimes Implies Function

- If two genes are similar, they are often homologous.
- If two genes are homologous, they often (but not always) have a similar function.
- If two proteins are homologous, they often have similar structure.
- Similarity can tell us a lot.

Alignments

- There are three main types of changes that can occur to DNA over time:
    1. a substitution that replaces one character with another,
    2. an insertion that adds one or more characters at a position,
    3. a deletion that deletes one or more characters at a position.
- Insertions and deletions are far less common than mutations.
- We use *gaps* in our alignment to reflect insertions and deletions.
- Note that gaps do not occur in nature; they are an artifact of aligning sequences.

Alignments

We can show an alignment in the following format:

Example

```
ACAGTAG
AC--TCG
```

- Here the first, second, fifth, and seventh positions align correctly.
- The sixth position is a *mismatch* which might have resulted from a **substitution**.
- The third and fourth positions are *gaps* which could equally well represent an **an insertion** into the first sequence, or a **deletion** from the second.

Alignments

- Letters that did not change line up on top of each other.
- Substitutions are written with one letter on top, and the substituted letter on the bottom.
- Insertions are written with the inserted part on either the top or bottom, with gaps along the other, so each can continue after that segment.

Alignments

- If we know what changes occurred throughout evolution to change one sequence into another, an alignment could be built to reflect that.
- As we do not usually know that information, we instead investigate **all possible alignments** (or a lot of them), and pick which one seems the most realistic.
- To do this, we need to assign each alignment a score, because it is very easy to compare scores and choose the best one.
- Then we need to pick scores that most accurately reflect likely homology.

First Technique: Simple Gap Penalty

- How do we assign a score to each alignment?
- We need to develop some type of *scoring function*, which will assign an integer to each possible alignment.
- We could develop a strategy like the following:

Simple Gap Penalty

Strategy

- every time we have a match, we add n to our score – the *match score*.
- every time we have a mismatch, we add m to our score – the *mismatch score*.
- every time we have a gap, we add g to our score – the *gap penalty*.

Simple Gap Penalty

Example

Consider the following alignment:

```
ACAGTAG
AC--TCG
```

- Assume a match score is 3, a mismatch score of 0 and a gap penalty of -2.
- Then this alignment would have a score of $3 + 3 - 2 - 2 + 3 + 0 + 3 = 8$.

Notice that in the above example, an alignment must have at least two gaps.

Open and Extension Penalties

- It is more likely to have a multiple adjacent gaps than multiple non-adjacent ones.
- This represents insertions and deletions of sequences of length longer than one.
- So we can break the gap penalty into two parts: an *open penalty* and a *extension penalty* to reflect this.

Open and Extension Penalties

We will likely assign a smaller extension penalty than an open penalty.

Example

- So with the alignment that we had above,
  ```
  ACAGTAG
  AC--TCG
  ```
- If the match score is still 3, a mismatch score of 0, an origination gap penalty of -2 and a length penalty of -1, then the score would be $3 + 3 - 2 - 1 + 3 + 0 + 3 = 9$.

Refining the Match and Mismatch Scores

- We've refined the gap penalty into an origination and length penalty.
- We would like to refine the match score and the mismatch score, as some substitutions are more common than others.
- We can construct a *scoring matrix* that reflects the individual scores associated with one character mutating into another.

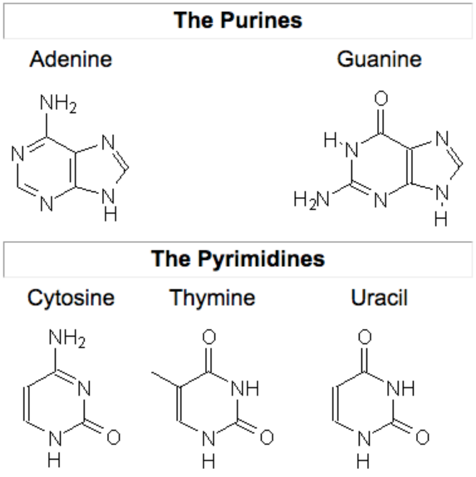
Scoring Matrix for BLAST

Example

Here is the scoring matrix used in BLAST.

	A	T	C	G
A	5	-4	-4	-4
T	-4	5	-4	-4
C	-4	-4	5	-4
G	-4	-4	-4	5

Refining the Match and Mismatch Scores



https://www.diffen.com/difference/Purines_vs_Pyrimidines

Refining the Match and Mismatch Scores

Example

Here is the scoring matrix which assigns a smaller penalty for substitutions in which a purine (A or G) is replaced by another purine and likewise for pyrimidine (C or T).

	A	T	C	G
A	1	-5	-5	-1
T	-5	1	-1	-5
C	-5	-1	1	-5
G	-1	-5	-5	1

Scoring Matrices

For proteins ...

- Commonly used scoring matrices for amino acid substitutions include the PAM (point accepted mutation) matrix and the BLOSUM matrix: the entries in both are based on the substitution rates observed between similar sequences.
- A matrix called BLOSUM62 is commonly used as a default
- More on protein scoring matrices later in this section.

Best Possible Alignment

- We would not only like to assign a score to each alignment, but we would like to find the *best possible* alignment.
- We could just try every possible alignment, calculate the score, then determine the best possible.
- There are a huge number of possible alignments (exponential in the lengths of the sequences).
- This approach would be take too long to do.

Best Possible Alignment

- We can use a more efficient algorithmic technique, called *dynamic programming*.

Definition – informal

Dynamic programming is a computational strategy which continuously breaks problems into smaller sub-problems, where the same sub-problems reappear. The strategy solves the sub-problems, storing the answers and using them to solve the larger problems.

- Dynamic programming has been applied to the *global sequence alignment* problem, that is, the alignment of two sequences along their entire length.

Global Sequence Alignment

The main algorithm used to solve this problem, is the Needleman-Wuncsh algorithm.

Definition

Given a sequence S, the length of S is denoted $|S|$.

Global Sequence Alignment

- The input is two sequences, S and R.
- We construct a $(|S| + 1) \times (|R| + 1)$ matrix whereby one sequence labels the rows and the other labels the columns
- We label the zeroth row and column with a gap symbol, '-'.

Global Sequence Alignment

Example

For example, if $S = ACAGTAG$ and $R = ACTCG$, then we get:

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

Global Sequence Alignment

Important point!

At entry (i, j) of the matrix, we are going to try to store the score of the optimal alignment of

$$S(1)S(2)\cdots S(i) \text{ and } R(1)R(2)\cdots R(j).$$

- Assume for now that we are using a fixed gap penalty, and match and mismatch scores.
- For this example, we let -1 be our gap penalty, $+1$ be our match score and 0 our mismatch score.

Global Sequence Alignment

- Like our dot plots, an alignment is represented by a path from the top-left corner to the bottom right corner.
- A vertical or horizontal move along the path would represent a gap, while a diagonal line would represent either a match or a mismatch.

Filling in Entries

basic idea

- Say we want to figure out the best possible way to align $S(1) \cdots S(i)$ with $R(1) \cdots S(j)$.
- If may look like any sort of alignment. There may be matches, mismatches and gaps in this alignment.

```
...S(1)    ...     S(i)
...R(1)    ...    R(j)
```

Algorithm Continued

There are three possible ways in which the last characters of $S(1) \cdots S(i)$ and $R(1) \cdots R(j)$ can align.

1. it ends with a gap in S,
   ```
   [some alignment of S(1) ... S(i)]   | ---
   [with              R(1) ... R(j-1)] | R(j)
   ```
2. it ends with a gap in R,
   ```
   [some alignment of S(1) ... S(i-1)]     | S(i)
   [with                  R(1) ... R(j)] | ---
   ```
3. it ends with either a match or a mismatch.
   ```
   [some alignment of S(1) ... S(i-1)]   | S(i)
   [with                R(1) ... R(j-1)]| R(j)
   ```

If we've already figured out the best way to align the blue parts, then we can figure out the best way to align $S(1) \cdots S(i)$ with $R(1) \cdots R(j)$ by considering just three possibilities.

Filling in Entries

- We start by filling in the first row and column with multiples of the gap penalty.

Example

In our example, if the gap penalty is -1, then we get

	-	A	C	T	C	G
-	0	-1	-2	-3	-4	-5
A	-1					
C	-2					
A	-3					
G	-4					
T	-5					
A	-6					
G	-7					

- This is because, for example, in order to align `ACTCG` with nothing, there will be 5 gaps in the second sequence.

Algorithm Continued

- We will continue by filling in the entry at position $(1, 1)$.
- If we can do this optimally, then it would represent the optimal alignment of the first letter of S (which is A) with the first letter of R (which is also A).
- We have three choices for this (and every) position.
- Either it ends with a gap in S, a gap in R, or a match/mismatch of the final characters of S and R.

Algorithm Continued

We can calculate the optimal such value by examining:

1. the value to the left, at position $(1, 0)$, and adding the gap penalty (a gap in R).
2. the value above, at position $(0, 1)$, and adding the gap penalty (a gap in S),
3. the value diagonal above and to the left, at position $(0, 0)$, and adding either the match score or the mismatch score depending upon whether the first character of S is equal to the first position of R.

Algorithm Continued

- By taking the max of these three possibilities, we are calculating the score of the optimal alignment of the first character of S with the first character of R.
- In the example, the entry at position $(1, 1)$ would be 1 since we have a match.
- The value is based on the value in $(0, 0)$.
- Now we can fill in the position $(1, 2)$ using exactly the same procedure, based on positions $(1, 1), (0, 2)$ and $(0, 1)$.
- This would represent the optimal score for aligning the first character of S with the first two characters from R.

The Example Continued

- We can then fill in the entire second row, and indeed the entire table using this procedure.
- By solving the small instances first, and recording the answers in the matrix, we can use those answers to compute the larger instances.
- If we had tried to solve the larger instances first, we would have calculated the smaller instances over and over again, for every possible alignment.

The Example Continued

the final table in our example is as follows:

	-	A	C	T	C	G
-	0	-1	-2	-3	-4	-5
A	-1	1	0	-1	-2	-3
C	-2	0	2	1	0	-1
A	-3	-1	1	2	1	0
G	-4	-2	0	1	2	2
T	-5	-3	-1	1	1	2
A	-6	-4	-2	0	1	1
G	-7	-5	-3	-1	0	2

The value in the bottom right entry (in our case, 2) represents the score of the optimal gapped alignment between the two sequences.

Formal Algorithm

- We understand how it works, however we would like to construct a real algorithm (in pseudocode).
- We will start with the construction of the matrix.
- For our pseudocode, a string is an array of characters.
- Although we haven't seen a matrix in our pseudocode, we will use an array with two coordinates rather than one coordinate.

Formal Algorithm

- So if **A** is a matrix, then $A(i,j)$ is the entry at the i^{th} row and the j^{th} column.
- Assume we have written a subroutine MAX(a, b, c) which takes three real numbers as input and returns the largest.
- Assume we have written a subroutine SCORE(s, r) which takes two characters, s and r, as input and returns the match score if they are identical and the mismatch score otherwise.
- Let m_+ be the match score, m_- the mismatch score and g the gap penalty.

Formal Algorithm

input: strings S and R
returns: The score of the optimal gapped alignment of S and R.

```
NEEDLEMAN_WUNSCH(S, R){
1   for $i \leftarrow 0$ to $|\mathbf{S}|$ do // initialization of zeroth column
2       $A(i,0) \leftarrow i * g$
3   for $j \leftarrow 0$ to $|\mathbf{R}|$ do // initialization of zeroth row
4       $A(0,j) \leftarrow j * g$
5   for $i \leftarrow 1$ to $|\mathbf{S}|$ do //fills in the rest of the array
6       for $j \leftarrow 1$ to $|\mathbf{R}|$ do
7           $A(i,j) \leftarrow$ MAX($A(i-1,j) + g, A(i,j-1) + g,$
                  $A(i-1,j-1)+$ SCORE($S(i-1), R(j-1)$)
                  )
8   return $A(|\mathbf{S}|, |\mathbf{R}|)$
}
```

Time Complexity of Needleman-Wunsch

Let's see how efficient this algorithm is

- In each statement the actual number of operations is a small constant times the number of array assignment operations.
- Array assignment predominate.
- For simplicity, we will count only array assignment operations.
- Recall that for determining computational complexity, constants are not significant.

Time Complexity of Needleman-Wunsch

- Line 1 and line 2 performs $|\mathbf{S}| + 1$ assignments. Line 3 and line 4 performs $|\mathbf{R}| + 1$. This totals $|\mathbf{S}| + |\mathbf{R}| + 2$.
- Assume that the MAX and SCORE subroutines take some small number of steps each. They are each performed once for each assignment in line 7.
- How many times will the assignment in line 7 be performed?
- Lines 6 and 7 are performed S times, once for each value of i in line 5.
- For each value of i, the assignment in line 7 is performed $|R|$ times, once for each value of j, as specified in line 6.
- Thus, the assignment in line 8 is performed $|S| \times |R|$ times.

Time Complexity of Needleman-Wunsch

- Line 1 and line 2 performs $|\mathbf{S}| + 1$ assignments. Line 3 and line 4 performs $|\mathbf{R}| + 1$. This totals $|\mathbf{S}| + |\mathbf{R}| + 2$.
- Assume that the MAX and SCORE subroutines take some small number of steps each. They are each performed once for each assignment in line 7.
- How many times will the assignment in line 7 be performed?
- Lines 6 and 7 are performed S times, once for each value of i in line 5.
- For each value of i, the assignment in line 7 is performed $|R|$ times, once for each value of j, as specified in line 6.
- Thus, the assignment in line 8 is performed $|S| \times |R|$ times.

Time Complexity of Needleman-Wunsch

- Line 1 and line 2 performs $|\mathbf{S}| + 1$ assignments. Line 3 and line 4 performs $|\mathbf{R}| + 1$. This totals $|\mathbf{S}| + |\mathbf{R}| + 2$.
- Assume that the MAX and SCORE subroutines take some small number of steps each. They are each performed once for each assignment in line 7.
- How many times will the assignment in line 7 be performed?
- Lines 6 and 7 are performed S times, once for each value of i in line 5.
- For each value of i, the assignment in line 7 is performed $|R|$ times, once for each value of j, as specified in line 6.
- Thus, the assignment in line 8 is performed $|S| \times |R|$ times.

Time Complexity of Needleman-Wunsch

- Line 1 and line 2 performs $|\mathbf{S}| + 1$ assignments. Line 3 and line 4 performs $|\mathbf{R}| + 1$. This totals $|\mathbf{S}| + |\mathbf{R}| + 2$.
- Assume that the MAX and SCORE subroutines take some small number of steps each. They are each performed once for each assignment in line 7.
- How many times will the assignment in line 7 be performed?
- Lines 6 and 7 are performed S times, once for each value of i in line 5.
- For each value of i, the assignment in line 7 is performed $|R|$ times, once for each value of j, as specified in line 6.
- Thus, the assignment in line 8 is performed $|S| \times |R|$ times.

Time Complexity of Needleman-Wunsch

- Thus, the subroutine will perform $|S| + |R| + 2 + |S| \times |R|$ assignment operations.

- This algorithm is $O(|\mathbf{R}||\mathbf{S}|)$.

The Example Continued

- That's great and all... we've got the score of the optimal alignment...
- but we want to **see** an alignment that gives us this score.
- We can compute it from our matrix.
- We want to find a path from the lower right corner to the upper left corner that corresponds to the maximum used to calculate each score of the path.

The Example Continued

- Starting from the lower right corner, we see which of either 1) the entry to the left 2) the entry above or 3) the entry diagonally left and above could have led to this score.

Example

In the lower right corner, we have a 2. The score to the left of that is a 0 which would not represent a valid path, since a gap would subtract 1, not add 2. Similarly for above. So the path must be diagonally left and above.

The Example Continued

- Starting from the lower right corner, we see which of either 1) the entry to the left 2) the entry above or 3) the entry diagonally left and above could have led to this score.

Example

In the lower right corner, we have a 2. The score to the left of that is a 0 which would not represent a valid path, since a gap would subtract 1, not add 2. Similarly for above. So the path must be diagonally left and above.

The Example Continued

A valid path is shown in bold from right to left:

		A	C	T	C	G
	0	-1	-2	-3	-4	-5
A	-1	**1**	0	-1	-2	-3
C	-2	0	**2**	1	0	-1
A	-3	-1	**1**	2	1	0
G	-4	-2	**0**	1	2	2
T	-5	-3	-1	**1**	1	2
A	-6	-4	-2	0	**1**	1
G	-7	-5	-3	-1	0	**2**

The reverse path is $\nwarrow\nwarrow\uparrow\uparrow\nwarrow\nwarrow\nwarrow$.

Thus we moved $\searrow\searrow\downarrow\downarrow\searrow\searrow\searrow$.

Constructing an Alignment

- Again, $\searrow\searrow\downarrow\downarrow\searrow\searrow\searrow$
- We can construct the alignment from left to right. The first arrow is $\searrow$, which indicates either a match or a mismatch, so we align the two characters:
  ```
  A
  A
  ```
- Similarly for the next position. Then we have a vertical arrow, which means we have a gap in the sequence labeling the columns, *R*. We get:
  ```
  ACA
  AC-
  ```
- We finish with
  ```
  ACAGTAG
  AC--TCG
  ```

Constructing an Alignment

- Again, $\searrow\searrow\downarrow\downarrow\searrow\searrow\searrow$
- We can construct the alignment from left to right. The first arrow is $\searrow$, which indicates either a match or a mismatch, so we align the two characters:
  ```
  A
  A
  ```
- Similarly for the next position. Then we have a vertical arrow, which means we have a gap in the sequence labeling the columns, R. We get:
  ```
  ACA
  AC-
  ```
- We finish with
  ```
  ACAGTAG
  AC--TCG
  ```

Constructing an Alignment

- Again, $\searrow\searrow\downarrow\downarrow\searrow\searrow\searrow$

- We can construct the alignment from left to right. The first arrow is $\searrow$, which indicates either a match or a mismatch, so we align the two characters:
```
A
A
```

- Similarly for the next position. Then we have a vertical arrow, which means we have a gap in the sequence labeling the columns, R. We get:
```
ACA
AC-
```

- We finish with
```
ACAGTAG
AC--TCG
```

Constructing an Alignment

- Again, $\searrow\searrow\downarrow\downarrow\searrow\searrow\searrow$
- We can construct the alignment from left to right. The first arrow is $\searrow$, which indicates either a match or a mismatch, so we align the two characters:
  ```
  A
  A
  ```
- Similarly for the next position. Then we have a vertical arrow, which means we have a gap in the sequence labeling the columns, R. We get:
  ```
  ACA
  AC-
  ```
- We finish with
  ```
  ACAGTAG
  AC--TCG
  ```

Constructing an Alignment

- There may be more than one path from the lower right corner to the upper left corner.
- If we are interested in just one alignment, then arbitrarily picking one path is enough.

Needleman-Wunsch Online

- Go to http://emboss.bioinformatics.nl/cgi-bin/emboss/needle.
- Paste in two FASTA sequences such as the myoglobin gene in human and in rat[12].
- By default, for DNA, this uses a match score of 5, and a mismatch score of -4. (We will explain scores used for proteins later).
- It finds the alignment that gets the highest score, even with separate gap open and gap extension penalty (we only studied the algorithm with simple gap penalty, but it can be extended).

[12] We'll discuss how to access these sequences later, but get the sequences here: http://www.ncbi.nlm.nih.gov/nuccore/44955887?report=fasta and here: http://www.ncbi.nlm.nih.gov/nuccore/48976077?report=fasta.

Semi-Global Alignment

- Needleman-Wunsch is very harsh on deciding whether two sequences are similar if one is significantly shorter than the other.
- For example, what if we have a short sequence and we want to see if it is similar to some part of a genome?
- In that case, Needleman-Wunsch would always give a horrible score due to the massive number of gaps because of the large length difference.

Semi-Global Alignment

- Let's say that we are given $S = GGACCTGG$ and $R = ACCT$, and we have a match score of 1, a mismatch score of 0 and a gap penalty of -1.
- In a sense, these two align almost perfectly, as R is a substring of S.
- We would like to modify the algorithm so as to not penalize gaps at the beginning or the end of the sequence.

Modification

$S = GGACCTGG$ **and** $R = ACCT$**, we have a match score of** 1**, a mismatch score of** 0 **and a gap penalty of** -1**.**

		A	C	C	T
	0	-1	-2	-3	-4
G	-1	0	-1	-2	-3
G	-2	-1	0	-1	-2
A	-3	-1	-1	0	-1
C	-4	-2	0	0	0
C	-5	-3	-1	1	0
T	-6	-4	-2	0	2
G	-7	-5	-3	-1	1
G	-8	-6	-4	-2	0

This score does not reflect the fact that these sequences are quite similar.

Modification

- The first row and column is used to penalize gaps at the beginning of sequences.
- That is why the entries are initialized to multiples of the gap penalty.
- If we initialize the first row and column to zero, then we are not penalizing for gaps at the beginning!

Modification

Modified matrix:

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

That score is better!

Modification

- There is still a problem with that modification, since we are still being punished for gaps at the end of sequences.
- After we reach the end of one sequence, gaps should be “free”.
- That is, when we reach the last row, we don’t charge a penalty for horizontal moves.
- Likewise, in the last column, we don’t charge a penalty for vertical moves.

Modification

Next modified matrix:

$$
\begin{array}{c|c|c|c|c|c|}
 & & A & C & C & T \\
\hline
 & 0 & 0 & 0 & 0 & 0 \\
\hline
G & 0 & 0 & 0 & 0 & 0 \\
\hline
G & 0 & 0 & 0 & 0 & 0 \\
\hline
A & 0 & 1 & 0 & 0 & 0 \\
\hline
C & 0 & 0 & 2 & 1 & 0 \\
\hline
C & 0 & 0 & 1 & 3 & 2 \\
\hline
T & 0 & 0 & 0 & 2 & 4 \\
\hline
G & 0 & 0 & 0 & 0 & 4 \\
\hline
G & 0 & 0 & 0 & 0 & 4 \\
\hline
\end{array}
$$

That score is best! It is reflecting the similarity between S and R.

Local Sequence Alignment

- Say we have two sequences, $S = TTTACCTGGG$ and $R = GGGACCTAAA$. There is a common pattern $ACCT$.
- In general, the common pattern would be longer and could be a conserved functional or structural domain.
- Global sequence alignment will not give this a great score, since the *entire* sequence of S is not that similar to the *entire* sequence of R.
- Semi-global sequence alignment will not give a great score since there is not a *subsequence* of one of the sequences that is similar to the *entire* other sequence.

Local Sequence Alignment

- It is a *subsequence* of one sequence that is similar to a *subsequence* of the other.
- We want to consider all possible subsequences of each sequence, consider all possible alignments of all pairs of subsequences, and identify the one with the best score.
- For that reason, we need to modify our global sequence algorithm once more, to capture this intuition.

Local Sequence Alignment

- This is exactly the purpose of local sequence alignment.
- The algorithm is known as the Smith-Waterman Algorithm.
- We only need to modify the global sequence alignment slightly to achieve this goal.

Local Sequence Alignment

Key ideas:

- once again construct a matrix storing the score between optimal alignments between prefixes of the two sequences being aligned
- the empty sequence is a subsequence of any sequence
- at any point in construction of the matrix, can also consider the alignment two empty subsequences

Smith-Waterman

- At every step, when we are filling in a given entry of the matrix, if the value is less than zero, we set the entry to zero instead. Assume match is 1, mismatch is -1, gap is -2.

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

Local Sequence Alignment

- To find the best matching subsequence between the two pairs, we work backwards, constructing a path, as with global sequence alignment.
- However, instead of starting at the lower right corner and working our way backwards to the upper left corner, we start at the *largest number* and work are way up and to the left until we hit a zero.

Smith-Waterman

Here is the resulting path:

		G	G	G	A	C	C	T	A	A	A
	0	0	0	0	0	0	0	0	0	0	0
T	0	0	0	0	0	0	0	1	0	0	0
T	0	0	0	0	0	0	0	1	0	0	0
T	0	0	0	**0**	0	0	0	1	0	0	0
A	0	0	0	0	**1**	0	0	0	2	1	1
C	0	0	0	0	0	**2**	1	0	0	1	0
C	0	0	0	0	0	1	**3**	1	0	0	0
T	0	0	0	0	0	0	1	**4**	2	0	0
G	0	1	1	1	0	0	0	2	3	1	0
G	0	1	2	2	0	0	0	0	1	2	0
G	0	1	2	3	1	0	0	0	0	0	1

Local Alignment

- To construct the alignment, we write the pairs, but we don't include the pairs that have zero in the entry.
- Our entry is
  ```
  ACCT
  ACCT
  ```
- This represents the "best" alignment of two subsequences of both sequences.

Time Complexity of Smith-Waterman

- An implementation still will performs $|S| + |R| + 2 + |S| \times |R|$ assignment operations.
- The constant (accounting for operations in calculating the maximum of four values, for example) is slightly larger
- This algorithm is still $O(|\mathbf{R}||\mathbf{S}|)$.

Smith-Waterman online

- There is an online version of the Smith-Waterman algorithm.
- `http://emboss.bioinformatics.nl/cgi-bin/emboss/water`.
- Separate gap opening and extension penalties can be chosen for this as well.

Gap Penalties

- We have used a simple gap penalty in presenting the Needleman-Wunsch and Smith-Waterman algorithms .
- We can use the more sophisticated gap penalty scheme (gap opening + gap extension) presented earlier.
- We need three matrices rather than one, but the computational complexity remains $O(|\mathbf{R}||\mathbf{S}|)$

Multiple Sequence Alignment

- We have only tried to align one sequence against another sequence.
- What if we have more than two sequences, and we want to see their similarities.
- We have k sequences, and would like to find the optimal sequence alignment between the k sequences, where k is potentially bigger than 2.

Multiple Sequence Alignment

- Can we extend Needleman-Wunsch and Smith-Waterman to work for k sequences?
- YES! Instead of using a 2-dimensional array, we use a k-dimensional array.
- For both Needleman-Wunsch and Smith-Waterman, we took the two sequences and labelled the rows and columns with them.
- Think of the rows as the y-axis and the columns as the x-axis.
- Restating the above: For both Needleman-Wunsch and Smith-Waterman, we took the two sequences and plotted them on the x- and y-axis.

Multiple Sequence Alignment

- Here, we plot each sequence on its own axis... axis $x_1, \ldots, x_k$.
- Runs in time $O(n^k)$, where n is the length of the longest sequence and k is the number of sequences.
- Will work for small values of k, but not practical for large k.

Sum of Pairs

- How to score a multiple sequence alignment?
- One technique used to score alignments is sum-of-pairs.
- Consider the sum of *column scores* for the alignment.
- Each column score is the sum of scores for all pairs within a column of the alignment.

Example

Alignment of the three sequences `AACTT`, `CATT`, `CAT` using a scoring function where a pair of gap characters scores 0, and otherwise a pair of characters `a`, `b` scores 1 if `a` and `b` are identical and -1 otherwise.

Sum of Pairs

Example

So with the alignment:

```
AACTT
CA-TT
CA-T-
```

in the first column, we get a score of $(-1) + (-1) + 1 = -1$, the second column, we get $1 + 1 + 1 = 3$, the third we get $(-1) + (-1) + 0 = -2$, the fourth we get $1 + 1 + 1 = 3$ and the last we get $1 + (-1) + (-1) = -1$. The total score for this alignment would be $-1 + 3 + (-2) + 3 + (-1) = 2$.

Profiles

Aside:

- Start with a multiple sequence alignment.
- A profile indicates how many times each letter occurs at each position

```
  A A C T T     the multiple sequence alignment
  C A - T T
  C A - T -
===========
A 1 3 0 0 0     the profile
C 2 0 1 0 0
T 0 0 0 3 2
G 0 0 0 0 0
```

Multiple Sequence Alignment

- Most multiple sequence alignment algorithms don't find the best alignment.
- It's too computationally difficult.

Multiple Sequence Alignment

- Instead they use a *heuristic*.
- Heuristic methods are used to speed up the process of finding a satisfactory solution, where an exhaustive search is impractical.
- We will only align two sequences at a time.
- Start with most closely related sequences first.

Multiple Sequence Alignment

- This technique is called *progressive alignment*.
- Input is n sequences, assumed to be related through evolution, as given by a phylogentic tree.
- Sometimes we are aligning a sequence to an alignment, or two alignments.
- The multiple alignment constructed is not guaranteed to be optimal, but is useful in practice.

Multiple Sequence Alignment

- Consider n homologous sequences: they are related through evolution, as given by a phylogenetic tree.
- If we knew the tree, then we could start by aligning the most related sequences.
- Then work our way "up" the tree by aligning hypothetical ancestors.

Multiple Sequence Alignment

- Algorithmically, this type of algorithm is called a *greedy algorithm*.
- A greedy breaks down problems into smaller pieces, and then chooses the best solution for that piece without considering the solution to the overall problem.

Greedy Algorithms

A cashier giving back change

- a person would like to pay for a product which costs x dollars and cents.
- their goal is to pay y where they want to minimize the number of bills and coins they get back from cashier. The bills and coins total $y - x$.

greedy algorithms

- A greedy algorithm might
    1. if the amount left to pay is less than or equal to 0, stop;
    2. select the smallest bill/coin greater than or equal to x that you have if you have one and give that, then stop;
    3. if you don't such a bill/coin, then you select the biggest bill/coin less than x that you do have, and give that to the cashier. Deduct the value of the bill or coin from x, and continue from step 1.
- Will this always find the best answer?

Greedy Algorithms

- Say the total comes to \$2.09 and you've got a \$10 bill and a nickel and four pennies. (Pretend pennies are still in active circulation.)
- How would the greedy algorithm pay? And what would the cashier give back?
- What would be the optimal answer?

ClustalW

- This is one of the most commonly used multiple sequence alignment program.
- It uses progressive alignment.
- We'll see quite a bit more on phylogenetic trees later, as they are closely related with multiple sequence alignment.

ClustalW - a Brief Overview

input: sequences $w_1, \ldots, w_k$

1. calculates the distance between every pair of sequences (using a pairwise alignment).

2. Creates an estimated phylogenetic tree (more on this in the next chapter) called a guide tree from the distance values.

3. Using the tree, it aligns closely related sequences using Needleman-Wunsch.

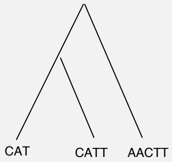
4. Each new alignment gives a sequence profile.

5. Sequences or profiles are aligned with other sequences or profiles "up the tree".

ClustalW - an Example

Example

- We are aligning AACTT, CATT, CAT.
- Say the guide tree looks like



ClustalW - an Example

Example

- The algorithm starts by aligning CAT with CATT using pairwise alignment.
- Perhaps:
  ```
  CAT-
  CATT
  ```
- Then it makes a profile based on that alignment:
  ```
    CAT-
    CATT
  ======
  A 0200
  C 2000
  T 0021
  G 0000
  ```

ClustalW - an Example

Example

- Then it takes that profile and “aligns” it with the remaining sequence AACTT.
- So, align AACTT to

```
CAT-
CATT
```

- Depending on the scoring function, it might yield

```
--CAT-
--CATT
AAC-TT
```

- This would be described by the profile

```
A 110200
C 003000
T 000032
G 000000
```

ClustalW

- ClustalW is very accurate for very similar sequences.
- However, inaccuracies are compounded through more comparisons.
- If you make a bad choice (mistake?) in an “early” alignment, it gets worse as you add in more sequence.
- This is why it is a greedy algorithm.

Database Searching

- We know that the Needleman-Wunsch and Smith-Waterman algorithms run in time $O(|S||R|)$.
- This is fine if we are comparing just a single gene to another gene.
- If we are trying to compare a gene against an entire database, this becomes impractical.
- As of August, 2018 there were over 208 million sequences in GenBank, with a total length of 260 billion bases.

Database Searching

- Introduce the FASTA (short for “fast-all”, not to be confused with the FASTA file format) and BLAST (basic local alignment search tool).
- FASTA was the first widely used program for similarity searching in databases.
- BLAST is used more now and is usually faster.

Database Searching

- Both are more complex than Smith-Waterman and Needleman-Wunsch, and based on heuristics.
- They do *not* guarantee optimal results, as Needleman-Wunsch and Smith-Waterman do, but they are more efficient.
- It doesn't necessarily find the best hits in a database, and the hits it does find aren't necessarily aligned optimally.
- They only work for local sequence alignment.

Key Idea

- Key idea in both: any “good” alignment will contain a few stretches where identical words subsequences (words) appear in both sequences.
- Look for such instances of identical words in both sequences, extending the best to full alignments.

BLAST

- BLAST stands for *Basic Local Alignment Search Tool*.
- It was created in 1990 at NCBI.
- It's main purpose is to compare a protein or DNA sequence to sequences in the various NCBI databases.
- It will create local sequence alignments of a sequence to those in the databases.

BLAST

- It allows to take a sequence and find a similar sequence in an entire organism's genome, or other organisms.
- You "submit" a sequence (the *query*).
- BLAST performs pairwise sequence alignments against an entire database (the *target*).
- It will report/produce local alignments for the sequences BLAST thinks match closest to your sequence.

Variants

Here are some variants of BLAST used for different purposes that we can pick.

Variants of BLAST

- **blastp**: a protein query sequence and a protein sequence database as target.
- **blastn**: a DNA query sequence and a DNA sequence database as target.
- **blastx**: a DNA query sequence is translated in all reading frames, and used to search a protein database.
- **tblastn**: a protein query sequence is used to search a DNA sequence database translated in all reading frames.
- **tblastx**: a DNA query sequence is translated to protein sequences in all reading frames, and the latter are used to search against a DNA database translated in all reading frames.

BLAST Website

- Visit BLAST here: `http://blast.ncbi.nlm.nih.gov`
- It is also possible to install BLAST on a local computer where there are even more advanced options that are not available through the website.
- It can search a database on your computer, or can run on your computer but search NBCI's database.

Need to Provide:

For every search, we need to pick:

- a BLAST program variant (blastp, blastn, etc.) to be used,
- a query sequence, by entering an accession number or pasting a sequence (or FASTA file contents) into a BLAST input box,
- a database to search against,
- any optional parameters.

Selecting the Database and Filters

- The Database can be selected.
- The default is non-redundant protein sequences, which combines together non-redundant sequences from GenBank, PDF, SwissProt and others.
- We can also restrict to the proteins in RefSeq, or other databases that we will discuss in Chapter 4.

Other Search Parameters

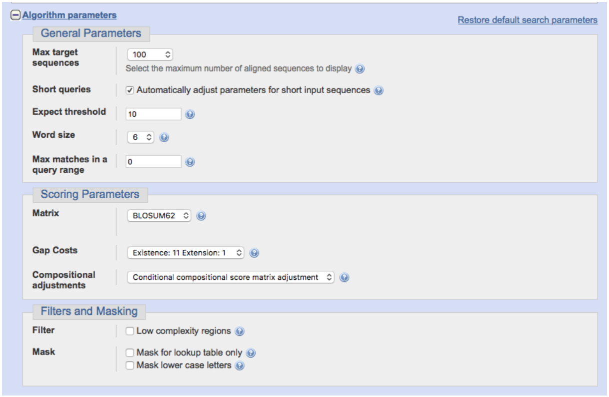
- For both blastp and blastn, we can (optionally) restrict to an organism, or a set of organisms.

- Start typing in ‘Homo sapiens’, ‘bacteria’, ‘eukaryota’, ‘mammalia’, ‘rodentia’, etc.

- Alternate algorithms in the ‘Program Selection’ section will be (in part) described later.

Some Advanced Parameters

Clicking "Algorithm parameters" will give a number of advanced parameters that can be set. These are the ones in blastp.



Algorithm parameters

Restore default search parameters

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	6
Max matches in a query range	0

Scoring Parameters

Matrix	BLOSUM62
Gap Costs	Existence: 11 Extension: 1
Compositional adjustments	Conditional compositional score matrix adjustment

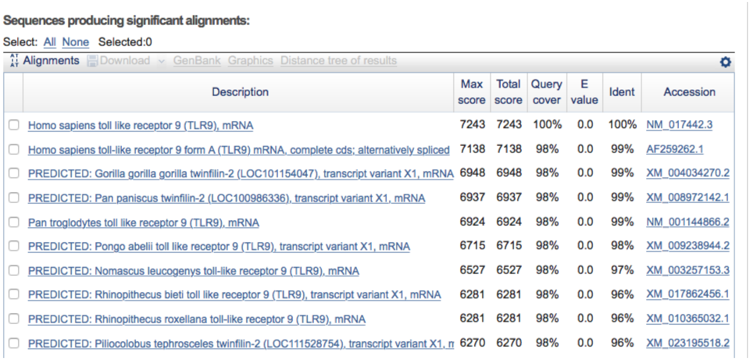
Filters and Masking

Filter	☐ Low complexity regions
Mask	☐ Mask for lookup table only ☐ Mask lower case letters

Using blastn

- Here's the TLR9 mRNA in human: `http://www.ncbi.nlm.nih.gov/nuccore/327365332?report=fasta`
- Then perform a blastn search to see 'similar' hits.

Description



Sequences producing significant alignments:

Select: All None Selected:0

Alignments Download GenBank Graphics Distance tree of results

	Description	Max score	Total score	Query cover	E value	Ident	Accession
☐	Homo sapiens toll like receptor 9 (TLR9), mRNA	7243	7243	100%	0.0	100%	NM_017442.3
☐	Homo sapiens toll-like receptor 9 form A (TLR9) mRNA, complete cds; alternatively spliced	7138	7138	98%	0.0	99%	AF259262.1
☐	PREDICTED: Gorilla gorilla gorilla twinfilin-2 (LOC101154047), transcript variant X1, mRNA	6948	6948	98%	0.0	99%	XM_004034270.2
☐	PREDICTED: Pan paniscus twinfilin-2 (LOC100986336), transcript variant X1, mRNA	6937	6937	98%	0.0	99%	XM_008972142.1
☐	Pan troglodytes toll like receptor 9 (TLR9), mRNA	6924	6924	98%	0.0	99%	NM_001144866.2
☐	PREDICTED: Pongo abelii toll like receptor 9 (TLR9), transcript variant X1, mRNA	6715	6715	98%	0.0	98%	XM_009238944.2
☐	PREDICTED: Nomascus leucogenys toll-like receptor 9 (TLR9), mRNA	6527	6527	98%	0.0	97%	XM_003257153.3
☐	PREDICTED: Rhinopithecus bieti toll like receptor 9 (TLR9), transcript variant X1, mRNA	6281	6281	98%	0.0	96%	XM_017862456.1
☐	PREDICTED: Rhinopithecus roxellana toll-like receptor 9 (TLR9), mRNA	6281	6281	98%	0.0	96%	XM_010365032.1
☐	PREDICTED: Piliocolobus tephrosceles twinfilin-2 (LOC111528754), transcript variant X1, m	6270	6270	98%	0.0	96%	XM_023195518.2

- The first hit is exactly the same sequence (same accession number!) as the query.
- It's in the database, so it was found.
- Not surprisingly, it got 100% identity.

Search Again

- Hit ‘Edit and Resubmit’.
- Perform the search again, but restrict the database to RNA from RefSeq.
- Do it again but find the best hit in ‘bovine’.
- Restricting the database to one like RefSeq, and restricting to an organism or taxon is a good way to cut down on unwanted results.

BLAST

- BLAST first takes the query sequence (the input), and breaks it into “words” of a fixed length (6 for proteins and 11 or 28 for nucleotides).
- All such words are calculated by “sliding a window” of that length.
- We will consider the problem for proteins.

BLAST Continued!

- for simplicity, we will consider a word size of 3 (a possible option for blastp)
- For each word of length 3, BLAST wants to see what sequences in the database have that word.
- Furthermore, BLAST looks at words similar to each word of length 3.
- For each word x of length 3, BLAST calculates *the neighbourhood* of x, which is the set of all words of length 3 which are “similar” to x.

BLAST continued!

- To determine which ones are similar, we match x to every word of length 3 using some scoring matrix.
- If they match with some score over the *neighbourhood score threshold T*, then we keep them.
- The set of all words of length 3 over the threshold are the *high-scoring words*.

How Does it Work?

Example

- Words are dealt with one at a time.
- Say BLAST is considering *RDQ*.
- Some other words are similar to *RDQ*.
- For example, the other words to the right are ranked from "most similar" to "least similar" with *RDQ*.
- The scores are calculated using the scoring matrix.

sequence	score
RDQ	16
RBQ	14
RDZ	14
KDQ	13
RDE	13
QDQ	12
...	

How Does it Work?

Example

- There is a parameter T that tells BLAST which of these words are "similar enough" to *RDQ* to require further investigation.
- If T is 13, then the first 5 are good. These are called *high scoring words*.
- Other lower words are ignored.
- T cannot be changed in the NCBI BLAST implementation but downloadable version has it.

sequence	score
RDQ	16
RBQ	14
RDZ	14
KDQ	13
RDE	13
QDQ	12
...	

How Does it Work?

- Then for the entire query, there are a lists of high scoring words of length 3.
- BLAST keeps an index of its entire database (GenBank) that allows it to find all occurrences of these words in the database very rapidly.
- In the original version of BLAST, if there was one hit with a sequence in the database, it pursued it further.
- It has been since refined to need two hits within a certain distance from each other. This speeds it up. We will ignore the second hit in our discussion.

BLAST continued!

- BLAST searches for the high-scoring words amongst each sequence in the database.
- Each match gives a so-called *seed*, which is a very short alignment.

BLAST continued!

- The seed is extended both to the right and left.
- The extension continues until the score starts to drop off.
- Once it drops off more than some threshold X, the extension is stopped and the alignment is trimmed back to the maximum point.
- If the score is above some threshold S, then the alignment will be reported by BLAST.
- The resulting alignment is called a *high-scoring segment pair* (or shortly HSP).

How Does it Work?

Example

- Our query was *LPRDQDY*, and it formed a seed with *FPRDZDY*.

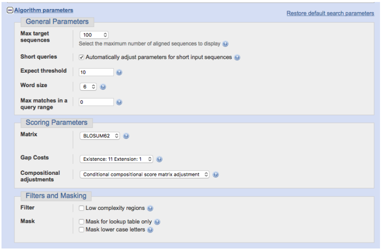
$$RDQ$$
$$RDZ$$

- Then BLAST extends the seed in both directions and continually calculates the score until it drops off more than X.

$$LP\,RDQDY$$
$$<------>$$
$$FP\,RDZDY$$

Some Advanced Parameters

- This explains the "Word size" parameter.
- For proteins, it can be set to 6 (the default), 3, or 2.
- Then 6 would be better at finding more close matches.
- It's harder to start a seed. Faster as a consequence.
- On the other hand, 2 would be better at finding more distant matches.
- It's easier to start a seed. Slower as a consequence.



Algorithm parameters

Restore default search parameters

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	6
Max matches in a query range	0

Scoring Parameters

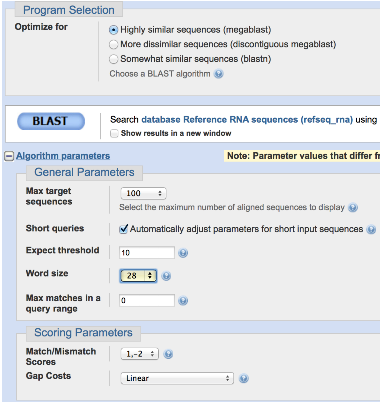
Matrix	BLOSUM62
Gap Costs	Existence: 11 Extension: 1
Compositional adjustments	Conditional compositional score matrix adjustment

Filters and Masking

Filter	☐ Low complexity regions
Mask	☐ Mask for lookup table only ☐ Mask lower case letters

Program Selection for Nucleotides

- For nucleotide BLAST, there are 3 options for 'Program Selection'.
- Default is Megablast, which is best for highly similar sequences (95% or higher).
- It uses a word size of 28.
- It's fastest.



Optimize for
- [✓ Highly similar sequences (megablast)]
- More dissimilar sequences (discontiguous megablast)
- Somewhat similar sequences (blastn)

Choose a BLAST algorithm

Search database Reference RNA sequences (refseq_rna) using

Show results in a new window

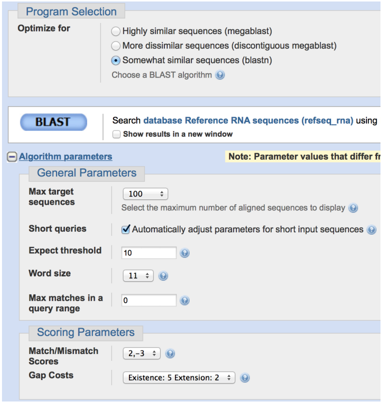
Note: Parameter values that differ fr

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
Max matches in a query range	0

Match/Mismatch Scores	1,-2
Gap Costs	Linear

Program Selection for Nucleotides

- The last option is ‘blastn’.
- It has default word size of 11, and can be lowered to 7.
- It’s slower, but can find more distant alignments.
- It also has different match/mismatch/gap scores.



Program Selection

Optimize for
- ( ) Highly similar sequences (megablast)
- ( ) More dissimilar sequences (discontiguous megablast)
- (●) Somewhat similar sequences (blastn)

Choose a BLAST algorithm

Search database Reference RNA sequences (refseq_rna) using

☐ Show results in a new window

Algorithm parameters

Note: Parameter values that differ fr

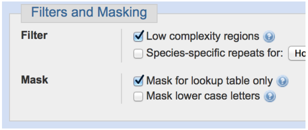
General Parameters

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	11
Max matches in a query range	0

Scoring Parameters

Match/Mismatch Scores	2,-3
Gap Costs	Existence: 5 Extension: 2

Low Complexity Regions


Filters and Masking

Filter
- ✓ Low complexity regions
- ☐ Species-specific repeats for: Ho

Mask
- ✓ Mask for lookup table only
- ☐ Mask lower case letters

- In nucleotide BLAST, there's a checkbox that's on for 'Low complexity regions'.
- This ignores sequences without much information content.
- For example, there can be repeated dinucleotides *CACACACACA* that might not be important in an alignment.

Return of the Word Game

- Let's return to the word game of earlier in this chapter.
- Everybody guessed or copied a word with 8 letters.
- If someone had mutations at 3 of 8 positions from my word, there was a $\frac{1}{238{,}659}$ chance that they did not copy.
- What if the class has $238,659$ people in it?
- Then, even if everyone came up with their word randomly (and are therefore not related to each other) we would **expect** someone to have 3 mutations off my word, simply by chance.

Return of the Word Game

- What if the class has $2 \times 238,659 = 477,318$ people in it?
- Then, even if everyone came up with their word randomly (and are therefore not related to each other) we would **expect** two persons to have 3 mutations off my word, simply by chance.

Key

- The key here is that the chance of finding a “pretty good” hit by chance is higher if it was searched against a big database.
- There’s a better chance of someone winning a lottery if a tonne of people have tickets.

Key

Key

- If we use BLAST to search a sequence against a database, our confidence that the alignments BLAST returns as best hits represent homology depends on the size of database it searched against.
- Indeed, the database is so huge, there might be “similar enough” sequences simply by chance, without any homology.

Key

- We need to adjust our confidence in homology by the size of the database.
- With each alignment, BLAST also provides an E Value.
- This is providing exactly such an adjustment.

E Value

- An E Value (Expect Value) provides a quantitative measure of whether returned alignments represent significant matches, or whether they occurred solely by chance.
- When BLAST returns an alignment with a score of S, it also returns an E Value.

E Value

The E Value is the expected number of different alignments with scores equivalent to or better than S to occur by chance in a database search.

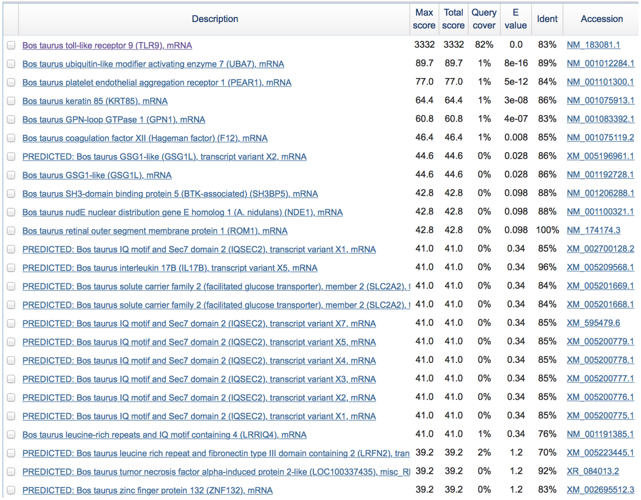
Example

- Say we submit a sequence and BLAST returns an alignment with a score of S and an E Value of 1.
- This means that in a database of this particular size, 1 match with a similar score is expected to occur by twice.
- “By chance” means “without any evolutionary relationship”, or “without any homology”.

E Value

- E Values over 1 are definitely too big to conclude there is homology, at least via sequence alignment alone.
- It has been estimated that 10^{-4} is an OK upper bound where there is likely homology.
- But lower than that is better.

Example



	Description	Max score	Total score	Query cover	E value	Ident	Accession
☐	Bos taurus toll-like receptor 9 (TLR9), mRNA	3332	3332	82%	0.0	83%	NM_183081.1
☐	Bos taurus ubiquitin-like modifier activating enzyme 7 (UBA7), mRNA	89.7	89.7	1%	8e-16	89%	NM_001012284.1
☐	Bos taurus platelet endothelial aggregation receptor 1 (PEAR1), mRNA	77.0	77.0	1%	5e-12	84%	NM_001101300.1
☐	Bos taurus keratin 85 (KRT85), mRNA	64.4	64.4	1%	3e-08	86%	NM_001075913.1
☐	Bos taurus GPN-loop GTPase 1 (GPN1), mRNA	60.8	60.8	1%	4e-07	83%	NM_001083392.1
☐	Bos taurus coagulation factor XII (Hageman factor) (F12), mRNA	46.4	46.4	1%	0.008	85%	NM_001075119.2
☐	PREDICTED: Bos taurus GSG1-like (GSG1L), transcript variant X2, mRNA	44.6	44.6	0%	0.028	86%	XM_005196961.1
☐	Bos taurus GSG1-like (GSG1L), mRNA	44.6	44.6	0%	0.028	86%	NM_001192728.1
☐	Bos taurus SH3-domain binding protein 5 (BTK-associated) (SH3BP5), mRNA	42.8	42.8	0%	0.098	88%	NM_001206288.1
☐	Bos taurus nudE nuclear distribution gene E homolog 1 (A. nidulans) (NDE1), mRNA	42.8	42.8	0%	0.098	88%	NM_001100321.1
☐	Bos taurus retinal outer segment membrane protein 1 (ROM1), mRNA	42.8	42.8	0%	0.098	100%	NM_174174.3
☐	PREDICTED: Bos taurus IQ motif and Sec7 domain 2 (IQSEC2), transcript variant X1, mRNA	41.0	41.0	0%	0.34	85%	XM_002700128.2
☐	PREDICTED: Bos taurus interleukin 17B (IL17B), transcript variant X5, mRNA	41.0	41.0	0%	0.34	96%	XM_005209568.1
☐	PREDICTED: Bos taurus solute carrier family 2 (facilitated glucose transporter), member 2 (SLC2A2), t	41.0	41.0	0%	0.34	84%	XM_005201669.1
☐	PREDICTED: Bos taurus solute carrier family 2 (facilitated glucose transporter), member 2 (SLC2A2), t	41.0	41.0	0%	0.34	84%	XM_005201668.1
☐	PREDICTED: Bos taurus IQ motif and Sec7 domain 2 (IQSEC2), transcript variant X7, mRNA	41.0	41.0	0%	0.34	85%	XM_595479.6
☐	PREDICTED: Bos taurus IQ motif and Sec7 domain 2 (IQSEC2), transcript variant X5, mRNA	41.0	41.0	0%	0.34	85%	XM_005200779.1
☐	PREDICTED: Bos taurus IQ motif and Sec7 domain 2 (IQSEC2), transcript variant X4, mRNA	41.0	41.0	0%	0.34	85%	XM_005200778.1
☐	PREDICTED: Bos taurus IQ motif and Sec7 domain 2 (IQSEC2), transcript variant X3, mRNA	41.0	41.0	0%	0.34	85%	XM_005200777.1
☐	PREDICTED: Bos taurus IQ motif and Sec7 domain 2 (IQSEC2), transcript variant X2, mRNA	41.0	41.0	0%	0.34	85%	XM_005200776.1
☐	PREDICTED: Bos taurus IQ motif and Sec7 domain 2 (IQSEC2), transcript variant X1, mRNA	41.0	41.0	0%	0.34	85%	XM_005200775.1
☐	Bos taurus leucine-rich repeats and IQ motif containing 4 (LRRIQ4), mRNA	41.0	41.0	1%	0.34	76%	NM_001191385.1
☐	PREDICTED: Bos taurus leucine rich repeat and fibronectin type III domain containing 2 (LRFN2), tran	39.2	39.2	2%	1.2	70%	XM_005223445.1
☐	PREDICTED: Bos taurus tumor necrosis factor alpha-induced protein 2-like (LOC100337435), misc_RI	39.2	39.2	0%	1.2	92%	XR_084013.2
☐	PREDICTED: Bos taurus zinc finger protein 132 (ZNF132), mRNA	39.2	39.2	0%	1.2	83%	XM_002695512.3

- Which E Values are significant?
- $8e - 16$ means $8 \cdot 10^{-16} = 8 \cdot \frac{1}{10{,}000{,}000{,}000{,}000{,}000}$.

Scoring Matrices

- The choice of scoring matrix has a large effect on blastp.
- DNA substitution matrices are less effective.
- Protein sequence comparisons can identify homology from organisms that last shared a common ancestor over 1 billion years ago [13].
- DNA sequence comparisons can only identify homology from up to 600 million years ago.

[13] WR Pearson, Effective protein sequence comparison. *Methods Enzymol.* 266, 227–258, 1996.

Scoring Matrices

- Some mutations have little effect on a protein's function, and so the corresponding entry in the matrix should be higher.
- An amino acid mutation is more likely to preserve a proteins function if the amino acids have similar biochemical properties.
- For example, the substitution of a polar with a polar amino acid, and nonpolar with nonpolar.

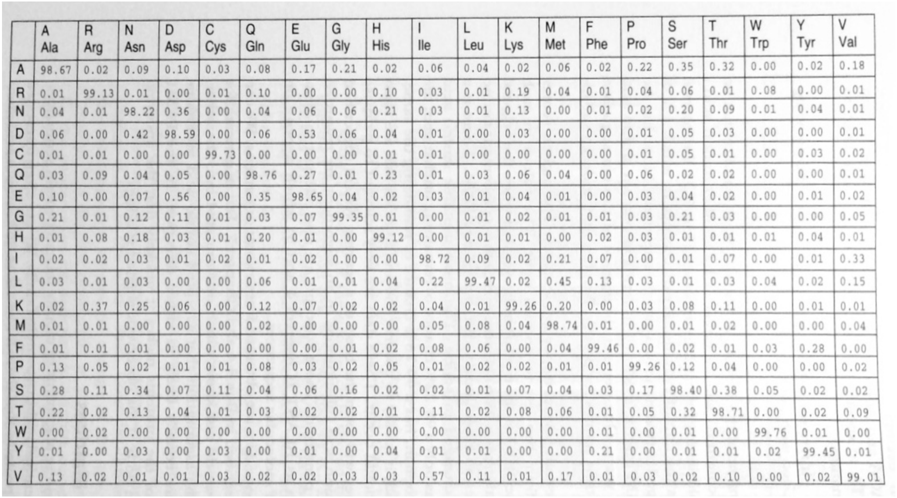
PAM

- An *accepted point mutation* is a replacement of an amino acid by another amino acid such that the entire species adopts that change as the predominant form of the protein.
- Essentially, it is a substitution of one amino acid with another, where the change takes.
- If the mutation causes harm, then it will likely not become predominant.

PAM

- Dayhoff considered 1572 changes in 71 groups of closely related proteins.
- They collected data on how often an amino acid changed into another amino acid, in sequences where 1% of the amino acids changed.
- So, they were collecting data when the sequences had not diverged that much.
- After that, they calculated the percentage of the time that amino acid x changes into amino acid y, for each x and y.

PAM1 Probability Matrix



	A Ala	R Arg	N Asn	D Asp	C Cys	Q Gln	E Glu	G Gly	H His	I Ile	L Leu	K Lys	M Met	F Phe	P Pro	S Ser	T Thr	W Trp	Y Tyr	V Val
A	98.67	0.02	0.09	0.10	0.03	0.08	0.17	0.21	0.02	0.06	0.04	0.02	0.06	0.02	0.22	0.35	0.32	0.00	0.02	0.18
R	0.01	99.13	0.01	0.00	0.01	0.10	0.00	0.00	0.10	0.03	0.01	0.19	0.04	0.01	0.04	0.06	0.01	0.08	0.00	0.01
N	0.04	0.01	98.22	0.36	0.00	0.04	0.06	0.06	0.21	0.03	0.01	0.13	0.00	0.01	0.02	0.20	0.09	0.01	0.04	0.01
D	0.06	0.00	0.42	98.59	0.00	0.06	0.53	0.06	0.04	0.01	0.00	0.03	0.00	0.00	0.01	0.05	0.03	0.00	0.00	0.01
C	0.01	0.01	0.00	0.00	99.73	0.00	0.00	0.00	0.01	0.01	0.00	0.00	0.00	0.00	0.01	0.05	0.01	0.00	0.03	0.02
Q	0.03	0.09	0.04	0.05	0.00	98.76	0.27	0.01	0.23	0.01	0.03	0.06	0.04	0.00	0.06	0.02	0.02	0.00	0.00	0.01
E	0.10	0.00	0.07	0.56	0.00	0.35	98.65	0.04	0.02	0.03	0.01	0.04	0.01	0.00	0.03	0.04	0.02	0.00	0.01	0.02
G	0.21	0.01	0.12	0.11	0.01	0.03	0.07	99.35	0.01	0.00	0.01	0.02	0.01	0.01	0.03	0.21	0.03	0.00	0.00	0.05
H	0.01	0.08	0.18	0.03	0.01	0.20	0.01	0.00	99.12	0.00	0.01	0.01	0.00	0.02	0.03	0.01	0.01	0.01	0.04	0.01
I	0.02	0.02	0.03	0.01	0.02	0.01	0.02	0.00	0.00	98.72	0.09	0.02	0.21	0.07	0.00	0.01	0.07	0.00	0.01	0.33
L	0.03	0.01	0.03	0.00	0.00	0.06	0.01	0.01	0.04	0.22	99.47	0.02	0.45	0.13	0.03	0.01	0.03	0.04	0.02	0.15
K	0.02	0.37	0.25	0.06	0.00	0.12	0.07	0.02	0.02	0.04	0.01	99.26	0.20	0.00	0.03	0.08	0.11	0.00	0.01	0.01
M	0.01	0.01	0.00	0.00	0.00	0.02	0.00	0.00	0.00	0.05	0.08	0.04	98.74	0.01	0.00	0.01	0.02	0.00	0.00	0.04
F	0.01	0.01	0.01	0.00	0.00	0.00	0.00	0.01	0.02	0.08	0.06	0.00	0.04	99.46	0.00	0.02	0.01	0.03	0.28	0.00
P	0.13	0.05	0.02	0.01	0.01	0.08	0.03	0.02	0.05	0.01	0.02	0.02	0.01	0.01	99.26	0.12	0.04	0.00	0.00	0.02
S	0.28	0.11	0.34	0.07	0.11	0.04	0.06	0.16	0.02	0.02	0.01	0.07	0.04	0.03	0.17	98.40	0.38	0.05	0.02	0.02
T	0.22	0.02	0.13	0.04	0.01	0.03	0.02	0.02	0.01	0.11	0.02	0.08	0.06	0.01	0.05	0.32	98.71	0.00	0.02	0.09
W	0.00	0.02	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.01	0.00	99.76	0.01	0.00
Y	0.01	0.00	0.03	0.00	0.03	0.00	0.01	0.00	0.04	0.01	0.01	0.00	0.00	0.21	0.00	0.01	0.01	0.02	99.45	0.01
V	0.13	0.02	0.01	0.01	0.03	0.02	0.02	0.03	0.03	0.57	0.11	0.01	0.17	0.01	0.03	0.02	0.10	0.00	0.02	99.01

Image from Pevsner, *Bioinformatics and Functional Genomics*.

Not a Scoring Matrix

- The matrix on the previous slide is *NOT* a scoring matrix.
- It is a set of probabilities.
- It doesn't make sense to use these probabilities as scores in a scoring matrix.
- We'll convert this to a scoring matrix later.

How They Were Calculated

- They counted the number of times that alanine changed to every other amino acid.
- Then they turned those into percentages.
- So, for example, in row 5, column 1, is the percentage of time that A changed into C (0.01% of the time). That means, out of the 10000 occurrences of A, it changed to C once.
- i.e. there was one alignment that looked like:

$$\begin{array}{ll} \cdots SAQ \cdots & \text{ancestor sequence} \\ \cdots SCQ \cdots & \text{protein analyzed by Dayhoff} \end{array}$$

Probability Matrix

- Also note that by looking at sequences that are 1% different, this is not a strict amount of time.
- The amount of time (in years) required for 1% of the amino acids to mutate, depends on the protein.
- Some proteins change much faster than others.
- Thus, this amount of time (in years) can be drastically different.

Another Kind of Generation

- But, we do not need to use time in years.
- We can just measure time in “PAM Generations”.

One PAM Generation

One PAM is the unit of evolutionary divergence in which 1% of the amino acids have been changed between two protein sequences.

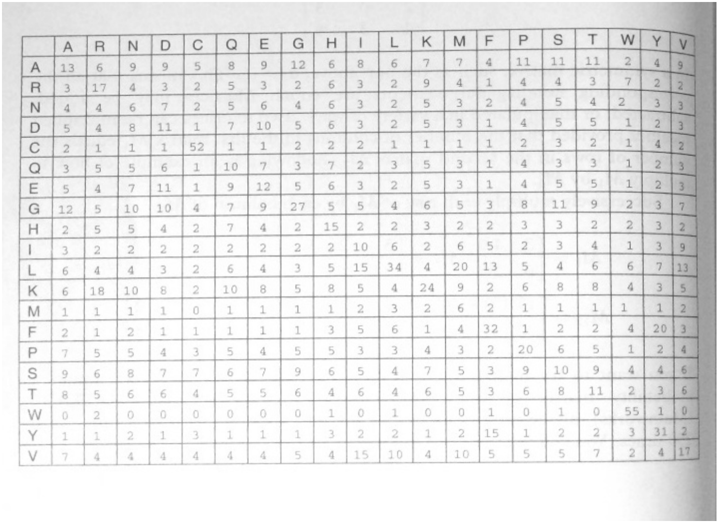
PAM 100?

- It makes sense to go past 100 PAM1 generations.
- Sequences that have diverged by more than 100 PAM1 generations are still homologous.
- It is ABSOLUTELY NOT the case that 100% of the amino acids will be different across 100 PAM1 generations.
- The amino acids that are more likely to mutate will likely mutate multiple times, while some of those that are less likely to mutate will likely not mutate.
- PAM100 would be appropriate for a protein of length 100 if there had been 100 mutations.

PAMx Probability Matrices

- Dayhoff et al. calculated the PAM1 probability matrix.
- They also gave a strategy for calculating other matrices, PAMx, for every positive integer x.
- PAMx contains the probabilities of each amino acid mutating into every other amino acid across x PAM1 generations.

PAM250 Probability Matrix



	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
A	13	6	9	9	5	8	9	12	6	8	6	7	7	4	11	11	11	2	4	9
R	3	17	4	3	2	5	3	2	6	3	2	9	4	1	4	4	3	7	2	2
N	4	4	6	7	2	5	6	4	6	3	2	5	3	2	4	5	4	2	3	3
D	5	4	8	11	1	7	10	5	6	3	2	5	3	1	4	5	5	1	2	3
C	2	1	1	1	52	1	1	2	2	2	1	1	1	1	2	3	2	1	4	2
Q	3	5	5	6	1	10	7	3	7	2	3	5	3	1	4	3	3	1	2	3
E	5	4	7	11	1	9	12	5	6	3	2	5	3	1	4	5	5	1	2	3
G	12	5	10	10	4	7	9	27	5	5	4	6	5	3	8	11	9	2	3	7
H	2	5	5	4	2	7	4	2	15	2	2	3	2	2	3	3	2	2	3	2
I	3	2	2	2	2	2	2	2	2	10	6	2	6	5	2	3	4	1	3	9
L	6	4	4	3	2	6	4	3	5	15	34	4	20	13	5	4	6	6	7	13
K	6	18	10	8	2	10	8	5	8	5	4	24	9	2	6	8	8	4	3	5
M	1	1	1	1	0	1	1	1	1	2	3	2	6	2	1	1	1	1	1	2
F	2	1	2	1	1	1	1	1	3	5	6	1	4	32	1	2	2	4	20	3
P	7	5	5	4	3	5	4	5	5	3	3	4	3	2	20	6	5	1	2	4
S	9	6	8	7	7	6	7	9	6	5	4	7	5	3	9	10	9	4	4	6
T	8	5	6	6	4	5	5	6	4	6	4	6	5	3	6	8	11	2	3	6
W	0	2	0	0	0	0	0	0	1	0	1	0	0	1	0	1	0	55	1	0
Y	1	1	2	1	3	1	1	1	3	2	2	1	2	15	1	2	2	3	31	2
V	7	4	4	4	4	4	4	5	4	15	10	4	10	5	5	5	7	2	4	17

Image from Pevsner, *Bioinformatics and Functional Genomics*.

PAMx

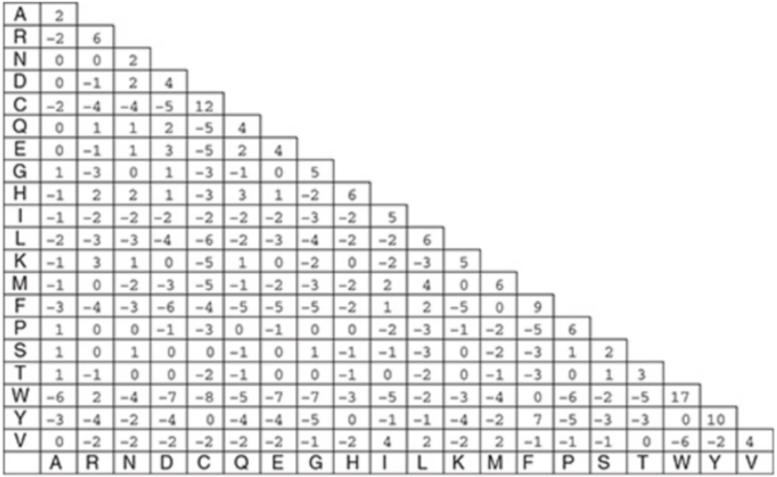
- After PAM1, all PAMx probability matrices are mathematically extrapolated.
- For that, matrix multiplication is used.

Turned Into Scores

- Each of the PAMx Probability Matrices can be turned into scoring matrices.
- It adjusts the probability of one amino acid changing into another by the probability of having the first amino acid at that position of the sequence in the first place.
- Also, it takes the ‘log’, allowing the scores to be added rather than multiplied, which is the way we’ve built our scoring schemes (to add the score at each position rather than multiply).

PAM250 Scoring Matrix

This is the PAM250 Scoring Matrix.



A	2																			
R	-2	6																		
N	0	0	2																	
D	0	-1	2	4																
C	-2	-4	-4	-5	12															
Q	0	1	1	2	-5	4														
E	0	-1	1	3	-5	2	4													
G	1	-3	0	1	-3	-1	0	5												
H	-1	2	2	1	-3	3	1	-2	6											
I	-1	-2	-2	-2	-2	-2	-2	-3	-2	5										
L	-2	-3	-3	-4	-6	-2	-3	-4	-2	-2	6									
K	-1	3	1	0	-5	1	0	-2	0	-2	-3	5								
M	-1	0	-2	-3	-5	-1	-2	-3	-2	2	4	0	6							
F	-3	-4	-3	-6	-4	-5	-5	-5	-2	1	2	-5	0	9						
P	1	0	0	-1	-3	0	-1	0	0	-2	-3	-1	-2	-5	6					
S	1	0	1	0	0	-1	0	1	-1	-1	-3	0	-2	-3	1	2				
T	1	-1	0	0	-2	-1	0	0	-1	0	-2	0	-1	-3	0	1	3			
W	-6	2	-4	-7	-8	-5	-7	-7	-3	-5	-2	-3	-4	0	-6	-2	-5	17		
Y	-3	-4	-2	-4	0	-4	-4	-5	0	-1	-1	-4	-2	7	-5	-3	-3	0	10	
V	0	-2	-2	-2	-2	-2	-2	-1	-2	4	2	-2	2	-1	-1	-1	0	-6	-2	4
	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V

Image from Pevsner, *Bioinformatics and Functional Genomics*.

Higher PAM Matrices

- Higher PAM numbers (such as PAM250) will do a better job at properly aligning sequences that are more dissimilar (distant).
- Amino acids that are unlikely to change have **much** higher match scores than those that are likely to change.
- Tryptophan match score is 17, and Alanine is 2.
- Many mismatches have a higher score than 2.

PAM

- PAM250 is useful even when proteins share approximately 20% amino acid identity.
- That said, alignment programs are not preset with the “best” scoring matrix, since that depends on how distant they are from each other.
- So how do we align proteins if we do not know the best scoring matrix?

BLOSUM

- BLOSUM is intended to be broadly useful – this helps solve the problem of not knowing the ideal scoring matrix.
- The BLOSUM (blocks substitution matrices) scoring matrices were created by Henikoff and Henikoff in 1992.
- BLOSUM62 was created using sequences sharing *at least* 62% identity.
- See `http://www.pnas.org/content/89/22/10915.full.pdf`

BLOSUM62

- BLOSUM62 merges together all proteins into an alignment such that all have 62% **or greater** identity.
- Because 62% or more is such a large range, it is useful in a variety of circumstances.
- Other variants of BLOSUM: BLOSUM50 use sequences that share at least 50% identity.

Which matrix to use

- As a default matrix, BLOSUM62 performs better overall than BLOSUM60 or BLOSUM70.
- That's why it is the default for so many alignment programs.
- But still, certain variants are better for highly similar alignments, and distantly related alignments.

BLOSUM

- Because the BLOSUM number represents the amount of percent identity, higher numbers are better for more similar sequences.
- Higher BLOSUM numbers (BLOSUM90) are better for very similar sequences.
- Lower BLOSUM numbers (BLOSUM50) are better for distantly related sequences.

Summarize

- PAM and BLOSUM work in opposite directions
- Lower PAM numbers are for more similar, higher BLOSUM numbers are for more similar.
- Higher PAM numbers are for more distant, lower BLOSUM numbers are for more distant.

best matrices

BLOSUM90	BLOSUM62	BLOSUM45
PAM30	PAM120	PAM250
less divergent	$\Longleftrightarrow$	more divergent

BLAST!

The parameters to BLAST really have an effect on it's output.

- The scoring matrix has a big effect on the set of high-scoring words.
- The word size has an effect.
- Increasing T will make the algorithm run faster, however we could miss some important alignments.
- Decreasing T will allow for noticing more distant relationships.
- Also, the X value.
- And the cutoff used for E Value.

Other Variations

- BLAST2Sequences is a variation, which is used to find alignments between any two protein or nucleotide sequences.
- PSI-BLAST - useful for studying distantly related proteins. A list of closely related proteins to the query sequence is constructed and merged into a profile (or position specific scoring matrix).
    - It is created by calculating scores for each position in the alignment. Highly conserved positions receive high scores, while weakly conserved positions will receive low scores.
    - This profile is then used to perform another BLAST search and the process iterates.