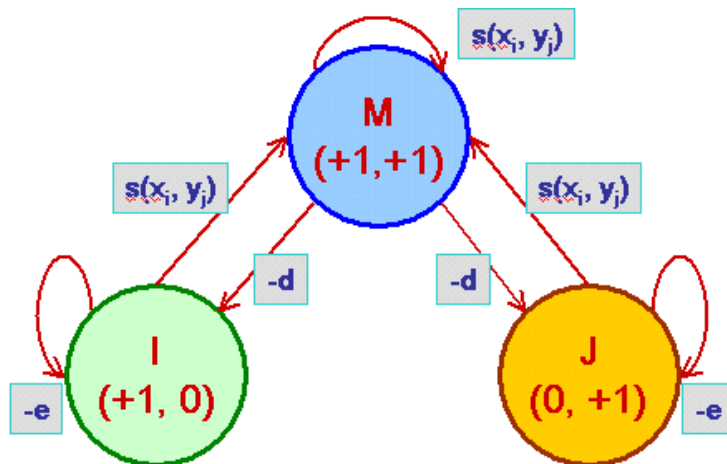


**CS 262: Computational Genomics**  
**Professor Serafim Batzoglou**  
**Lecture 8: Pair HMMs and Protein Alignment**

Scribed on February 2, 2006 by Ben Handy

**Finite State Automaton for Alignment**

We can create a state model that corresponds to generating an alignment between two sequences  $x$  and  $y$ . There is a one-to-one correspondence between alignments and strings composed of M, I, and J, which is also a path through the automaton, such that the sum of +1 terms for  $x$  is equal to the length of sequence  $x$  and the sum of +1 terms for  $y$  is equal to the length of sequence  $y$ .



```

-AGGCTATCACCTGACCTCCAGGCCGA--TGCCC---
TAG-CTATCAC--GACCGC-GGTCGATTTGCCCGACC
IMMJMMMMMMJJMMMMMJMMMMMMIIMMMMI I I

```

To simplify our model, we will make the restriction that gaps in  $x$  cannot immediately follow gaps in  $y$ , and vice versa. The model would work just as well if we allow this. This is a design decision – if you wish to have gaps in  $x$  followed by  $y$ , you can.

We have labeled every transition in the model with a score. Transitions to state M indicate letter-to-letter correspondences, so they are labeled with  $s(x_i, y_j)$  corresponding to the substitution score for replacing  $x_i$  with  $y_j$ . We know which  $i$  and  $j$  to use, based on the current sum of +1's for  $x$  and  $y$  thus far. We label every transition from M to a gap state (I or J) with the gap initiation penalty  $-d$ , and we label each transition from a gap state to itself with the gap extension penalty  $-e$ .

Every path through this model corresponds to an alignment. If we sum every score on the transitions of a path, we will have the same score as a global alignment dynamic programming problem with affine gap penalties.

## Dynamic Programming Review

We defined three matrices for our dynamic programming, one corresponding for each of our states M, I, J.

**M(i, j):** Optimal alignment of  $x_1 \dots x_i$  to  $y_1 \dots y_j$  ending in **M**

**I(i, j):** Optimal alignment of  $x_1 \dots x_i$  to  $y_1 \dots y_j$  ending in **I**

**J(i, j):** Optimal alignment of  $x_1 \dots x_i$  to  $y_1 \dots y_j$  ending in **J**

And our standard dynamic programming recurrences:

Initialization:

$$M(0,0) = 0;$$

$$M(i, 0) = M(0, j) = -\text{infinity, for } i, j > 0$$

$$I(i,0) = d + I * e; \quad J(0, j) = d + j * e$$

Iteration:

$$\mathbf{M(i, j)} = s(x_i, y_j) + \max \{ M(i - 1, j - 1), I(i - 1, j - 1), J(i - 1, j - 1) \}$$

$$\mathbf{I(i, j)} = \max \{ -e + I(i - 1, j), -d + M(i - 1, j) \}$$

$$\mathbf{J(i, j)} = \max \{ -e + J(i, j - 1), -d + M(i, j - 1) \}$$

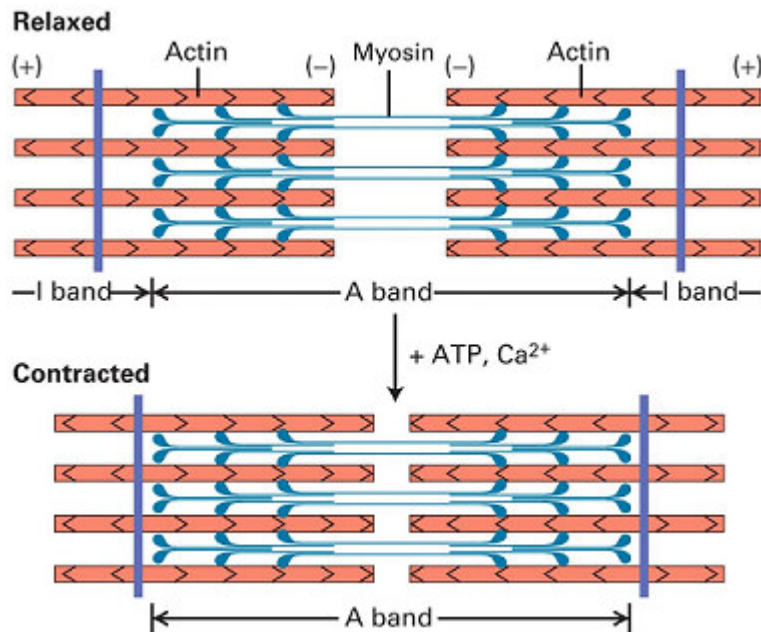
Termination:

The optimal alignment given by  $\max [ M(m, n), I(m, n), J(m, n) ]$

## Protein Sequence and Structure (consider taking CS273 next quarter for more details.)

Muscles very are important to the biology of vertebrates. Muscles are very long cells: they are 1-50 millimeters long, and 10-50 micrometers in diameter. They are composed of road-like structures (filaments) composed of two proteins: **Actin** and **Myosin**.

Actin filaments are like strings or roads, and myosin filaments are in between these, and can pull the actin filaments closer together. This allows the muscles to contract to 70% of their original length.



Actin is a very important protein that can be found in almost every cell (even bacteria). It is involved in forming road-like structures inside the cell. We tend to think of cells as bags of different chemicals and proteins and enzymes, but really cells are very elaborately sculptured machines, and actin plays a major roll of keeping the shape and structure of each cell so they can perform the appropriate functions. A road-like structure of actin is often an alternating pattern of three different types of actin.



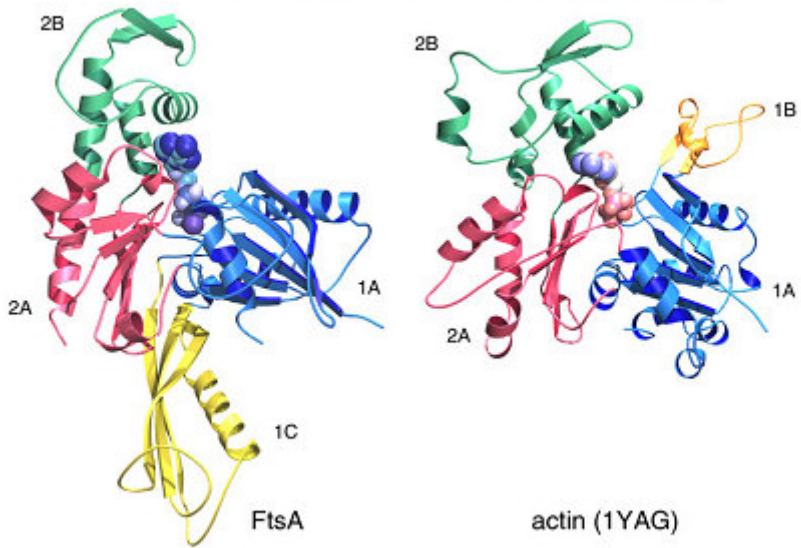
Actin is very ancient and very abundant. In fact, it is the most abundant protein in our bodies. There are 1-2 actin genes in bacteria, while humans have 6 Actin genes that differ by about 4 amino acids. Humans and amoebas have 80% similarity in actin proteins. Actin is generally grouped in the following categories:

**alpha-actin:** in muscles

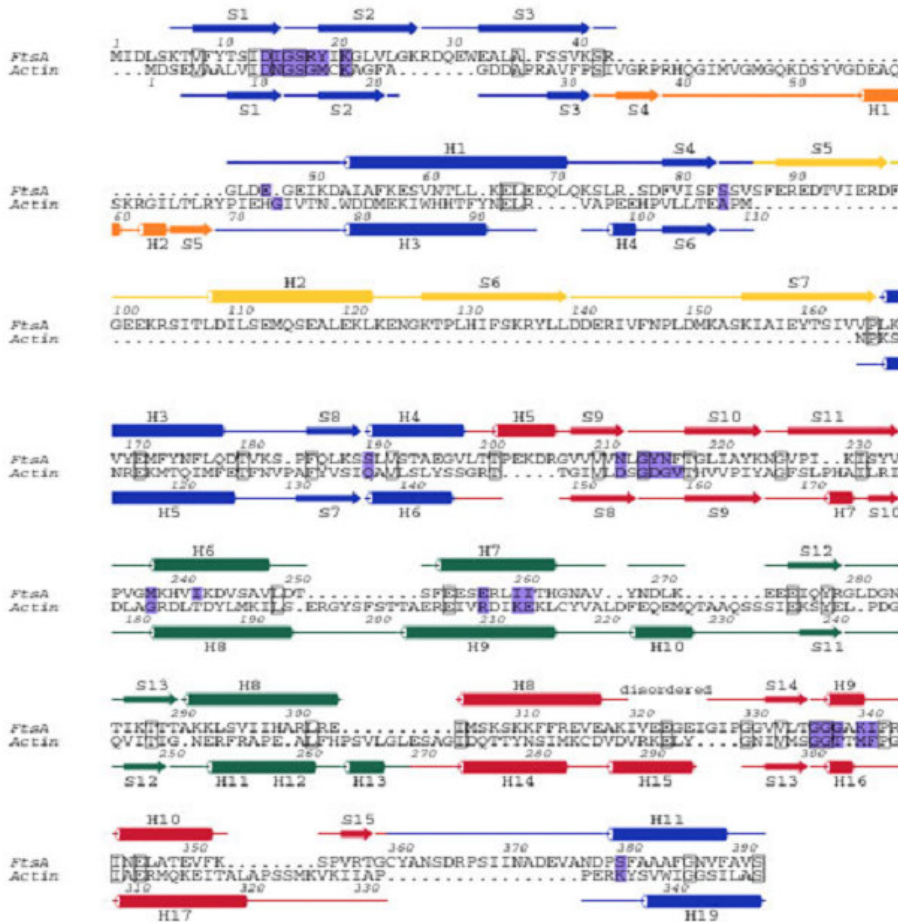
**beta-actin, gamma-actin:** in non-muscle cells.

Another extremely important protein is FtsZ (or FtsA) protein that helps in cell division. The FtsZ protein creates a ring around the dividing cell, and makes the ring smaller and smaller until the cell divides. If we compare the FtsA protein to actin, we see that they are very similar, sharing 3 main domains, but FtsA has an additional domain (shown in yellow below).

## FtsA is a member of the actin family



If we examine their sequences and align them, we see they have very high similarity in 3 domains, but the 4<sup>th</sup> domain acts as a large gap (in yellow below).

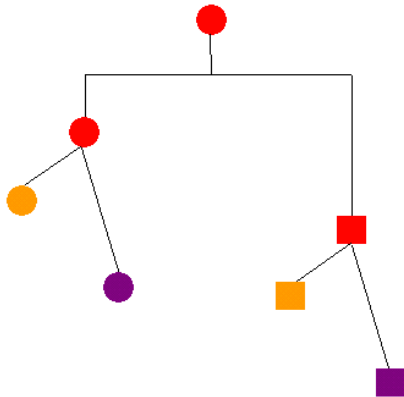


## Protein Phylogenies

There are 2 main ways that proteins evolve (families are formed):

1. **Speciation:** one ancestor species splits. When you compare the proteins, they are generally similar, but have some differences.
2. **Duplication:** A region is copied, so one area is free to take on a new purpose.

A phylogenetic tree can show how proteins are related. Duplication is generally represented by horizontal and vertical lines, while speciation is represented by direct edges with no bends. In the example below, the two yellow proteins are found in the same species, and the two purple proteins are found in the same species. Actin must be a billion years old, or older.



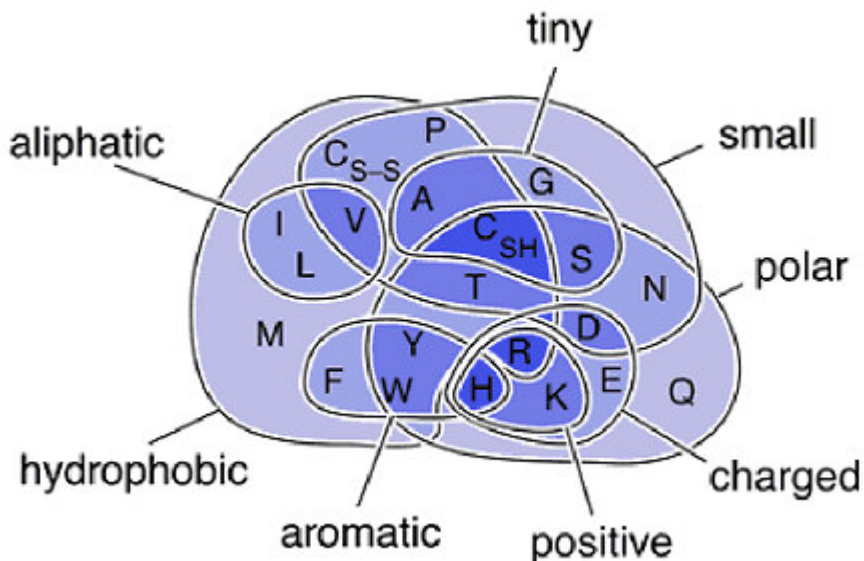
## Protein Structure

In proteins, structure determines function. Also, the sequence determines the structure. This means that we can discover the function of a protein, by only examining the sequence. One sequence has exactly one structure, and one structure generally has one function. Some examples of protein functions are: regulation of gene expression, structure of cell (actin), movement (muscle), catalysis, transport of molecules in and out of cells, signaling between cells, etc. We will not study structure much in this course, but it is interesting to note that studying sequences, and alignment of sequences, can be connected to structure and function of proteins.

Proteins are composed of amino acids that are connected with peptide bonds, and differ from each other only in the side chain of amino acids. There are 20 different amino acids, which have very different chemical properties and shapes, and can be divided into the following 3 categories:

1. **Hydrophobic:** These stay close to each other, and away from water (much like oil does). Cytoplasm is mostly water, which causes these to fold. Examples include: Alanine, Valine, Isoleucine, Leucine, Methionine, Phenylalanine, Tyrosine, Tryptophan.
2. **Hydrophilic:** (or polar). These like to stay close to water, forming bonds with it. Examples include: Lysine, Arginine, Histidine, Aspartate, Serine, Threonine, Glutamate, Asparagine, Glutamine.
3. **Special:** These like to join together, forming extremely strong sulphide bonds. Cysteine, Glycine, Proline.

There are other ways to divide amino acids, such as by shape. Amino acids can be similar to one another in a variety of ways, while being different in other ways. To see how this works, see the chart below. Also notice how some proteins are similar in many ways, while others have almost nothing in common.



Secondary structure of proteins has two major substructures:

1. **Alpha helixes** (roughly 3.6 amino acids per helical turn), stabilized by hydrogen bonds
2. **Beta strands/sheets**, also stabilized by hydrogen bonds.

Tertiary structure of proteins can be represented in several ways:

1. **Backbone Trace**: shows the space occupied by the backbone of the protein.
2. **Ball and Stick**: shows precise molecules, very high detail.
3. **Ribbons**: alpha helices are curly ribbons while beta strands are flat sheets.
4. **Solvent-Accessible Surface**: shows what parts a water molecule can reach (smoother, shows rough shape).

We are sequencing more and more genomes, finding more and more available proteins. We are also getting exponential growth in the number of available protein structures. The main protein database PDB shows 35,000 protein structures have been experimentally determined. However, we do not discover many new classes of structures (folds). About 15 years ago, Cyrus Chothia observed (based on statistics) that there should be roughly 1000 different shapes for proteins in nature. This has turned out to be quite accurate - 945 discovered so far, and new families are discovered rarely now. At the same time, it was estimated when several genomes would be completely sequenced and how many genes would be found in each. While the number of genes has been fairly accurate, we have been able to finish sequencing genomes much more quickly, because of progress in sequencing technology. It appears that our ability to sequence is growing exponentially. We are sequencing roughly 10 mammalian species per year now and this will probably continue to grow exponentially. If this continues, projects that sound very difficult, such as sequencing all human individuals, may not be as far off as we think.

Proteins are classified into folds at the highest level, which describe the rough shape of the protein. Within a fold there are several superfamilies, which are (perhaps distantly) evolutionarily related. Each superfamily has several families, which usually share some functionality. If two proteins have 25% sequence similarity tend to be in same family, and have similar structure.

There are several good sources for browsing protein classifications:

SCOP: <http://scop.berkeley.edu>, manual classification, probably the single best source (created by A Murzin, a genius who has all of the structures in his head).

CATH: <http://www.biochem.ucl.ac.uk/bsm/cath>, semi-manual classification (C. Orengo).

FSSP: <http://www.ebi.ac.uk/dali/fssp/fssp.html>, automatic classification (L. Holm).

Some example classes in SCOP:

- All alpha proteins
- All beta proteins
- alpha and beta proteins (alpha/beta)
- alpha and beta proteins (alpha+beta)
- Multi-domain proteins
- Membrane and cell surface proteins
- Small proteins
- Coiled coil proteins

## Amino Acid Substitution

Amino acids (the 20 we spoke of earlier) with similar chemical and shape characteristics are more interchangeable. This means we can substitute one for another without making a major change in the protein. To describe interchangeability in a more rigorous manner, we can examine BLOSUM matrices.

BLOSUM substitution matrices are a common method for protein alignment. The idea is to start with a database of curated, gap-free alignments (BLOCKS) and sequences with >X% similarity are placed in a cluster.

We then calculate:

$A_{ab}$ : the number of times the amino acid  $a$  is aligned with amino acid  $b$  in a different cluster, correcting by dividing by clusters sizes ( $1/m*n$ ).

And estimate:

$P(\mathbf{a}) = (\sum_b A_{ab}) / (\sum_{c \leq d} A_{cd}) = [\text{Number of substitutions involving } a, \text{ divided by total substitutions}]$

$P(\mathbf{a}, \mathbf{b}) = A_{ab} / (\sum_{c \leq d} A_{cd}) = [\text{Number of substitutions between } a \text{ and } b, \text{ divided by total substitutions}]$

We then find the correlation between  $a$  and  $b$  by dividing the two:

$P(\mathbf{a}, \mathbf{b}) / P(\mathbf{a})P(\mathbf{b}) > 1 \Rightarrow a \text{ and } b \text{ are correlated.}$

## Probabilistic Interpretation of Alignment

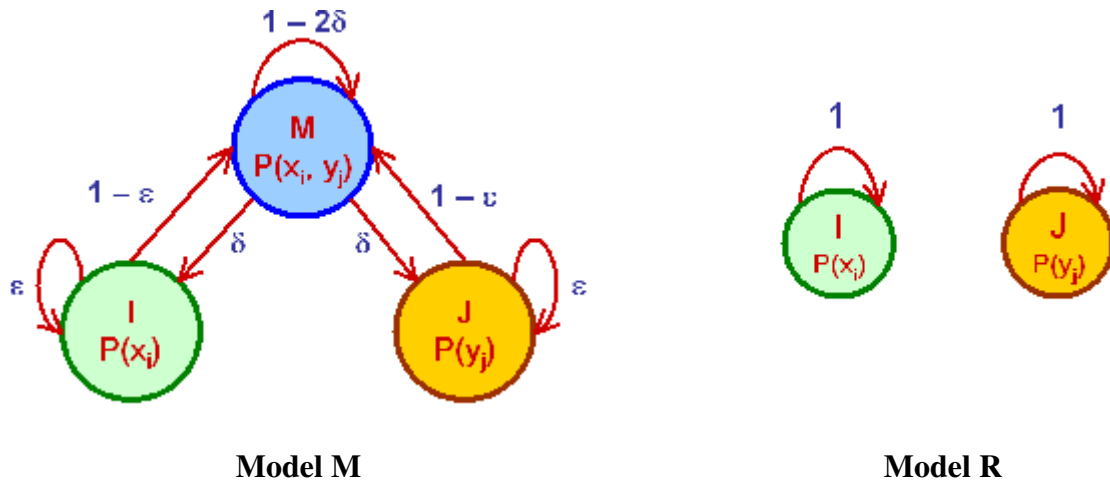
An alignment is a hypothesis that two sequences are related by evolution. We would like to find the *most likely* alignment rather than using arbitrary scoring parameters. We can then assert the likelihood that the sequences are related.

In the beginning of the lecture, we generated an alignment with a finite automaton. We will now consider this automaton as a pair hidden markov model that produces an alignment between two sequences (again we are ignoring possible transitions between I and J, or possible start and end states). We have three states M, I, and J, as shown in Model M on the left, below. Each state emits a pair of characters. State M emits one character from X and one character from Y. State I emits a character in X and a gap in Y. State J emits a gap in X and a character in Y. The probabilities of transition are described in the diagram (on edges) – delta affects how often gaps are opened, and epsilon affects how long gaps last. To be more precise:

**delta:** set so that  $1/2\delta$  is avg. length before next match

**epsilon:** set so that  $1/(1 - \epsilon)$  is avg. length of a gap

Emission probabilities are listed inside the nodes. The emission probability in M reflects likelihood of aligning  $x_i$  and  $y_j$ , which could be calculated as described using BLOSUM matrices. The emission probabilities in I and J reflect the composition of the sequences X and Y respectively. They model the likelihood of emitting each character from its string.



This model M assumes that the two sequences are related, since we generate them simultaneously. The model R (random model), shown on the right, above, handles the opposite assumption. It generates sequence X, and then it generates sequence Y independently. So the probability of generating a sequence is the product of emitting the letter for each letter in X, then multiplying by the same for Y.

$$P(x, y | R) = P(x_1) \dots P(x_m) P(y_1) \dots P(y_n) = P_i P(x_i) P_j P(y_j)$$

We can now compare the likelihood of each model, to see which fits the data better. We will first examine how each pair of letters contribute to the likelihood of the alignment:

**Model M** (these are a little off, as we ignore start and end gap costs)

$(1 - 2\delta) P(x_i, y_j)$  when aligned

$\epsilon P(x_i) P(y_j)$  when gapped

**Model R**

$P(x_i)P(y_j)$

Assuming that we use mostly matches and mismatches in Model M, and avoid gaps, we can focus on the first equation. Since  $(1-2\delta)$  is close to 1, Model M is close to just  $P(x_i, y_j)$  and so comparing our models reduces to examining the same quantity as we did for BLOSUM matrices:

$P(x_i, y_j) / P(x_i) P(y_j)$

To compare, we divide model M by Model R and take logarithms:

$$s(x_i, y_j) = \log P(x_i, y_j) / P(x_i) P(y_j) + \log (1 - 2\delta)$$

$$d = -\log \delta (1 - \epsilon) P(x_i) / (1 - 2\delta) P(x_i) = -\log \delta (1 - \epsilon) / (1 - 2\delta)$$

$$e = -\log \epsilon P(x_i) / P(x_i) = -\log \epsilon$$

The substitution score is a little off, because it does not account for the case that we have come to state M from I or J, with probability  $(1-\epsilon)$ , instead of from M, with probability  $(1-2\delta)$ . However, we have accounted for this in our gap initiation penalty so the total will be correct.

So the likelihood of the alignment is equivalent to multiplying all of the terms of the alignment model, and dividing by the terms of the random model. The maximum likelihood alignment is the alignment that will be produced by Needleman-Wunsch global alignment with affine gaps. Therefore, we have described an HMM where computing the maximum likelihood alignment is the same as computing global alignment with affine gaps, as we did in lecture 2. This means that running the Viterbi algorithm on a Pair HMM is equivalent to running Needleman-Wunsch dynamic programming. The equivalence makes sense when we examine the Viterbi recurrences:

$$V_M(i, j) = \max \{ V_M(i-1, j-1), V_I(i-1, j-1), V_J(i-1, j-1) \} + s(x_i, y_j)$$

$$V_I(i, j) = \max \{ V_M(i-1, j) - d, V_I(i-1, j) - e \}$$

$$V_J(i, j) = \max \{ V_M(i, j-1) - d, V_J(i, j-1) - e \}$$

Now by comparing the substitution score for a match with the substitution score for a mismatch, and taking into account the conservation or similarity between the two sequences, it is not hard to calculate the number of matches/mismatches that forms the border between a more likely alignment model and a more likely random model.