Computational Biology
Lecture 8: Substitution matrices
Saad Mneimneh

As we have introduced last time, simple scoring schemes like +1 for a match, -1 for a mismatch and -2 for a gap are not justifiable biologically, especially for amino acid sequences (proteins). Instead, more elaborated scoring functions are used. These scores are usually obtained as a result of analyzing chemical properties and statistical data for amino acids and DNA sequences.

For example, it is known that same size amino acids are more likely to be substituted by one another. Similarly, amino acids with same affinity to water are likely to serve the same purpose in some cases. On the other hand, some mutations are not acceptable (may lead to demise of the organism). PAM and BLOSUM matrices are amongst results of such analysis. We will see the techniques through which PAM and BLOSUM matrices are obtained.

Substrition matrices

Chemical properties of amino acids govern how the amino acids substitue one another. In principle, a substrition matrix s, where s_{ij} is used to score aligning character i with character j, should reflect the probability of two characters substituing one another. The question is how to build such a probability matrix that closely maps reality? Different strategies result in different matrices but the central idea is the same. If we go back to the concept of a high scoring segment pair, theory tells us that the alignment (ungapped) given by such a segment is governed by a limiting distribution such that

$$q_{ij} = p_i p_j e^{\lambda s_{ij}}$$

where:

- s is the subsitution matrix used
- q_{ij} is the probability of observing character i aligned with character j
- p_i is the probability of occurrence of character i

Therefore,

$$s_{ij} = \frac{1}{\lambda} \ln \frac{q_{ij}}{p_i p_j}$$

This formula for s_{ij} suggests a way to constrcut the matrix s. If high scoring alignments are to be real, $\{q_{ij}\}$ represent the desired probabilities of substitions, while $\{p_i\}$ represent the background probabilities of occurrence. By observing related families of sequences, one could estimate q and p and hence obtain the matrix s by using some scaling factor λ. Note that

$$\lambda \sum_{i,j} p_i p_j s_{ij} = \sum_{i,j} p_i p_j \ln \frac{q_{ij}}{p_i p_j} = \sum_{i,j} p_{ij} \ln \frac{q_{ij}}{p_{ij}}$$

Information theory tells us that the above sum is strictly less than 0 if $p \neq q$, which is a desired property. Also, note that the score S' in bits of a given segment (see previous lecture) is

$$S' = \log \frac{e^{\lambda S}}{K}$$

Since S is a sum of terms of the form $\frac{1}{\lambda} \ln \frac{q_{ij}}{p_i p_j}$, $e^{\lambda S}$ is a product of terms of the form $\frac{q_{ij}}{p_i p_j}$. Therefore, S' reflect the log likelihood of observing the alignment due to substitution (governed probabilistically by q) relative to simply by chance (governed probabilistically by p). Division by the constant K adjusts the score to account for the rate of observing maximal scoring segments as described in the previous lecture.

Here's another intuitive approach that justifies the above scheme. To construct a substitution matrix to score protein alignments, a family of proteins can be considered, and a multiple alignment of all the protein sequences in the family is obtained. Again, we are considering alignments with no gaps; therefore, we assume sequences have the same length (a valid assumption if they are related) and, therefore, the multiple alignment is trivial.

- for any pair of amino acids i and j we need q_{ij}, the probability of observing i aligned with j (which is same as q_{ji}), and
- p_i, the probability of observing an i.

The question is, in an alignment (ungapped) of sequences x and y that aligns two of their amino acids i and j, did this happen by chance or was indeed because of a mutation from i to j or vice versa? To capture this complementary behaviors, we consider two models:

- M, where x and y are related and obtained according to the joint probabilities q_{ij}, and
- R, where x and y are unrelated and obtained independently at random according to the individual probabilities p_i and p_j.

Considering this, now the score is the likelihood that the sequences are related compared relative to them being unrelated. This is called the *odds ratio* and is mathematically expressed as:

$$score(x, y) = \frac{P(x, y|M)}{P(x, y|R)} = \frac{\prod_i q_{x_i y_i}}{\prod_i p_{x_i} \prod_i p_{y_i}}.$$

This formula says that the score of the (ungapped) alignment is the probability that the symbols of x and y are aligned because they are related, relative to the probability of their symbols being aligned just by chance.

For two aligned amino acids i and j, if we take i's point of view, what is the probability to see a j on the other side? Well, this is the probability that an i mutated into a j, $p(i \to j)$. However, there is a mere chance of p_j for a random occurence of a j as well. Hence, the probability ratio $\frac{p(i \to j)}{p_j}$ reflects how much i believes that this j is related to it.

Now, since $q_{ij} = p_i p_{j|i} = p_i p(i \to j)$ (the probability of observing an i and its mutated form), we can also express the likelihood as:

$$\frac{q_{ij}}{p_i p_j} = \frac{p(i \to j)}{p_j}$$

.

By doing this for every pair of aligned symbols in the alignment and finding the product of the terms, we obtain the formula above, which reflect how much we believe that the two entire sequences are related.

In all the alignment algorithms we have seen so far, we relied on the fact that the score is additive, and this was a key property for the dynamic programming to work. In this case - when the score is computed as $\prod_{i,j} \frac{q_{x_i y_i}}{p_{x_i} p_{y_i}}$ - it is multiplicative. In order to make it additive we can take the log and compute: $\log \prod_{i,j} \frac{q_{x_i y_i}}{p_{x_i} p_{y_i}} = \sum_{i,j} \log \frac{q_{x_i y_i}}{p_{x_i} p_{y_i}}$. This is called the log-odds ratio. Therefore, the values making up the sum will be the individual scores found in the matrix, hence $s_{ij} = \log \frac{q_{ij}}{p_i p_j}$ up to some scaling factor.

Note that this is symmetric, so scoring i aligned with j is the same as scoring j aligned with i, hence, the direction of the alignment is not important (but one could in principle make a distinction if needed).

Now the important question: how to compute p_i, p_j, and q_{ij}? We're going to look at two ways of computing this: PAM and BLOSUM matrices.

PAM (Point Accepted Mutations) matrices

PAM stands for Point Accepted Mutations. An accepted mutation is defined as a mutation that was positively selected by the environment and did not caused the demise of the organism. A PAM matrix M holds the probability of i being replaced by j in a certain evolutionary time period. The longer the evolutionary period of time, the more difficult it is to determine the correct values. The reason being that i could mutate several times before becoming a j, and it will be hard to capture all these intermediate mutations, since we only observe i and j. What we are going to do is look over mutations that occurred in a relatively short evolutionary period of time. One unit of evolution is defined to be the amount of evolution that changes, on the average, 1 in 100 amino acids. Considering this unit, a 1-PAM matrix is first computed. Using this as a starting point, a k-PAM matrix can be generated from the 1-PAM matrix.

For a 1-PAM matrix M, M_{ij} is going to be $p(i \to j)$ scaled by a factor, such that the expected number of mutations is 0.01; in other words it is the same as having the probability of 1 in 100 for a mutation to occur. The computational steps that lead to the 1-PAM matrix are:

- Compute p_i for every i.
- Compute $p(i \to j)$ for every pair i and j and let $M_{ij} = p(i \to j)$.
- Scale M such that the expected number of mutations $\sum_i p_i(1 - M_{ii})$ is 0.01.
- Use $s_{ij} = 10 \log_{10} \frac{M_{ij}}{p_j}$ to obtain the additive scores. s_{ij} is rounded to an integer and here, the scaling factor 10 is used just to provide a better integer approximation.

Next we'll take a closer look at each of these steps. Let the frequency count f_{ij} be the number of times i is aligned with j counting both directions. Then, let the number of occurrences of i, $f_i = \sum_j f_{ij}$, and the count of all characters $f = \sum_i f_i$. Now, we can estimate $p_{ij} = \frac{f_{ij}}{f}$, whose meaning is simply the rate at which i was found to be aligned with j. Similarly, the rate of finding an occurence of i is $p_i = \frac{f_i}{f}$.

Now, having both p_{ij} and p_i determined, the elements of the matrix M are being computed as: $M_{ij} = p(i \to j) = \frac{p_{ij}}{p_i}$. M is indeed a probability matrix, and this can be proved by noting that $\sum_j M_{ij} = 1$.

To illustrate this step of computing a matrix M, let's have a quick example. Let the alignment be:

$$\begin{matrix} A & B \\ A & A \end{matrix}$$

In this case, the frequencies are:

$$\begin{aligned} f_{AB} &= 1 \\ f_{BA} &= 1 \\ f_{AA} &= 2 \\ f_A &= \textstyle\sum_X f_{AX} = f_{AB} + f_{AA} = 3 \\ f_B &= \textstyle\sum_X f_{BX} = f_{BA} = 1 \\ f &= \textstyle\sum_X f_X = 4 \end{aligned}$$

hence the estimated probabilities:

$$\begin{aligned} p_{AB} &= \tfrac{f_{AB}}{f} = \tfrac{1}{4} \\ p_{BA} &= \tfrac{f_{BA}}{f} = \tfrac{1}{4} \\ p_{AA} &= \tfrac{f_{AA}}{f} = \tfrac{1}{2} \\ p_A &= \tfrac{f_A}{f} = \tfrac{3}{4} \\ p_B &= \tfrac{f_B}{f} = \tfrac{1}{4} \\ p(A \to B) &= \tfrac{p_{AB}}{p_A} = \tfrac{1}{3} \\ p(B \to A) &= \tfrac{p_{BA}}{p_B} = 1 \end{aligned}$$

The matrix M will be:

$$M = \begin{bmatrix} \frac{2}{3} & \frac{1}{3} \\ 1 & 0 \end{bmatrix}$$

The expected number of mutations is

$$\sum_X p_X(1 - M_{XX}) = p_A(1 - M_{AA}) + p_B(1 - M_{BB}) = \frac{3}{4}(1 - \frac{2}{3}) + \frac{1}{4}(1 - 0) = 0.5 = 50\%$$

The next step is the scaling of M such that it is consistent with the definition of a 1-PAM matrix: 1 in 100 expected mutations. Suppose matrix M's elements, M_{ij}, are scaled by a factor $\alpha \neq 1$. In this case the new values become $M_{ij} = \alpha M_{ij}$. This will change the values of the row sums such that $\sum_j M_{ij} = \alpha \neq 1$. Since we want a probability matrix - every row sums up to 1 - a small adjustment is needed: we will add $1 - \alpha$ to every element on the main diagonal:

$$\begin{aligned} M'_{ij} &= \alpha M_{ij}, i \neq j \\ M'_{ii} &= \alpha M_{ii} + 1 - \alpha \end{aligned}$$

This will restore the property of a probability matrix. Now what should α be? Let's compute the new expected number of mutations:

$$\sum_i p_i(1 - M'_{ii}) = \sum_i p_i(1 - \alpha M_{ii} - 1 + \alpha) = \alpha \sum_i p_i(1 - M_{ii})$$

This is just α multiplied by the old expected number of mutations. Therefore, we can set α appropriately. For instance, in the example above, $\alpha = 0.02$.

Having a 1-PAM matrix computed, the question is how to compute a 2-PAM matrix? In other words, what is the probability $p_2(i \to j)$ of i mutating into j in two units of evolution. This is the probability of i mutating into k, for some k, in the first unit of evolution, and then, k mutating into j in the second unit of evolution. Mathematically, this can be expressed as:

$$p_2(i \to j) = \sum_k p(i \to k)p(k \to j) = \sum_k M_{ik}M_{kj}$$

This is the formula used to obtain the entry corresponding to the pair i and j when multiply M by itself. Hence, the 2-PAM matrix is just M^2. An analogous step is used to show that the k-PAM matrix is the same as M^k. When working with a k-PAM probability matrix the score will be computed in the same way: $s_{ij}^k = 10 \log_{10} \frac{M_{ij}^k}{p_j}$. The only change is that now the values of M^k are plugged instead of those of M.

BLOSUM (BLOCKS Substitution Matrices) matrices

As mentioned earlier, BLOSUM are another type of matrices used in scoring sequence alignments. They are intended to be used for scoring similarities of protein sequences that are evolutionary far apart (distant). Computing their values is done using the information stored in a database of blocks (called the BLOCKS database) where each block is a multiple ungapped alignment of related protein sequences. The sequences of each block are clustered, putting two sequences into the same cluster if their percentage of matching aligned residues - or level of similarity - is above a certain threshold $L\%$. We define two sequences to be *distant* if they fall in different clusters. Therefore, two distant sequences differ by at least $(100 - L)\%$. The computation of BLOSUM-L, for a particular value of L, is based on counting the number of mutations among *distant* sequences only. Therefore, lower values of L correspond to longer evolutionary times, and are applicable for more distant sequences.

As explained above, in computing a BLOSUM-L matrix's entries, we want to count the number of mutations between distant sequences only - the ones that are less than $L\%$ similar. The value f_{ab} is the relative frequency of seeing a aligned with b. Whenever such an alignment is observed for two sequences that are in different clusters, f_{ab} is incremented by $\frac{1}{n_1 n_2}$, where n_1 and n_2 are the sizes of the two clusters (we scale by the size of the cluster since larger clusters are more likely to contain mutations).

The steps through which a matrix is computed are:

- Estimate $p_i = \frac{\sum_j f_{ij}}{\sum_{k,l} f_{ij}}$;
- Estimate $q_{ij} = \frac{f_{ij}}{\sum_{k,l} f_{kl}}$;
- *BLOSUM-L(i,j)*$= \log \frac{q_{ij}}{p_i p_j}$ with some scaling factor λ.

Consider an example where sequences are generated at random (so we are not using the BLOCKS database here) such that $p_A = p_G = p_C = p_T = \frac{1}{4}$ and the level of similarity is 50%, i.e. the probability that two aligned residues are the same is 0.5. Then if $L = 50\%$, we expect to have one cluster, where:

$$p_{AA} = p_{GG} = p_{CC} = p_{TT} = 0.5\frac{1}{4} = \frac{1}{8}$$

and

$$p_{AG} = p_{AC} = p_{AT} = p_{GA} = p_{GC} = p_{GT} = p_{CA} = p_{CG} = p_{CT} = p_{TA} = p_{TG} = p_{TC} = 0.5\frac{1}{12} = \frac{1}{24}$$

Then a match will have a score

$$m = \log \frac{1/8}{1/4.1/4} = 1$$

and a mismatch will have a score

$$-s = \log \frac{1/24}{1/4.1/4} = -0.585$$

References

Setubal J., Meidanis, J., Introduction to Molecular Biology, Chapter 3.
Drubin R. et al., Biological Sequence Analysis, Chapter 2.