Mathematical Models, Algorithms, and Statistics of Sequence Alignment

By

Tatiana Aleksandrovna Orlova

Specialist
Saratov State University 2002

Submitted in Partial Fulfillment of the Requirements

for the Degree of Master of Science in

Mathematics

College of Arts and Sciences

University of South Carolina

2010

Accepted by:

Eva Czabarka, Director of Thesis

Joshua Cooper, Second Reader

James Buggy, Interim Dean of the Graduate School

Acknowledgments

I would like to thank Dr. Eva Czabarka and Dr. Joshua Cooper for their helpful advice and patience.

Abstract

The problem of biological sequence comparison arises naturally in an attempt to explain many biological phenomena. Due to the combinatorial structure and pattern preserving properties of the sequences it has attracted not only biologists, but also mathematicians, statisticians and computer scientists. In this work we study one of the most effective tools widely used for comparison of biological sequences - *sequence alignment*. We present the basic theory of sequence alignment from computational, biological, and statistical perspectives. We will also present and analyze results of computer simulations that effectively illustrate one possible application of this theory.

Contents

Acknowledgments . . . ii

Abstract . . . iii

List of Tables . . . v

List of Figures . . . vi

Chapter 1. Introduction: Where Mathematics Meets Biology . . . 1

- 1.1. The Organized Complexity of Life . . . 1
- 1.2. Biological Sequences and Their Patterns . . . 4
- 1.3. Biological Sequence Comparison . . . 9

Chapter 2. Sequence Alignment . . . 11

- 2.1. Pairwise Sequence Alignment . . . 11
- 2.2. Scoring Schemes . . . 14
- 2.3. Alignment Algorithms . . . 27

Chapter 3. Statistics of Local Sequence Alignment . . . 50

- 3.1. Hypothesis Testing . . . 50
- 3.2. Random Variables and Random Processes . . . 53
- 3.3. Ungapped Local Alignment Scores Statistics . . . 62
- 3.4. Gapped Local Alignment Scores Statistics . . . 78

Chapter 4. Conclusions . . . 87

Bibliography . . . 89

Appendix A. C++ Code . . . 94

List of Tables

Table 1.1 Amino acids and their abbreviations. 7

Table 3.1 Robinson & Robinson amino acid counts. 80

Table 3.2 Simulations results. 80

Table 3.3 Known estimates for λ and μ. 81

List of Figures

Figure 1.1 Central dogma of molecular biology 4

Figure 1.2 DNA chemical structure 5

Figure 1.3 DNA complementarity 5

Figure 1.4 Lambda repressor protein bound to a lambda operator DNA sequence. 8

Figure 2.1 BLOSUM62 substitution matrix 22

Figure 2.2 Alignment graph for sequences ATACTGG and GTCCGTG. . 28

Figure 2.3 Another alignment graph for ATACTGG and GTCCGTG. . . 29

Figure 2.4 Extended alignment graph. 33

Figure 2.5 Extended alignment network 45

Figure 2.6 The optimal global path. 46

Figure 2.7 Global alignment score matrix $(S(i, j))$ 47

Figure 2.8 The optimal local path. 48

Figure 2.9 Local alignment score matrix $(\hat{S}(i, j))$ 49

Figure 3.1 Ungapped alignment and its accumulative score. 63

Figure 3.2 Random walk corresponding to the ungapped local alignment. 66

Figure 3.3 Counts of large alignment scores, $L = 400$ 82

Figure 3.4 Probability density plot of simulated and estimated data for sequences of length L=40. 83

Figure 3.5 Cumulative probability distribution plot of simulated and estimated data for sequences of length $L = 40$. 83

Figure 3.6 Probability density plot of simulated and estimated data for sequences of length $L = 100$. 84

Figure 3.7 Cumulative probability distribution plot of simulated and estimated data for sequences of length $L = 100$. 84

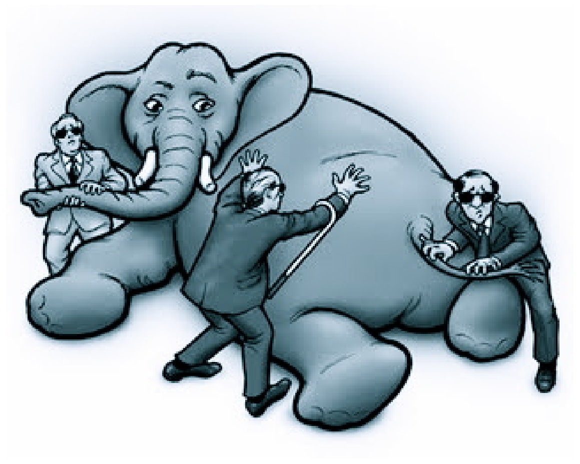
Figure 3.8 Probability density plot of simulated and estimated data for sequences of length $L = 200$. 85

Figure 3.9 Cumulative probability distribution plot of simulated and estimated data for sequences of length $L = 200$. 85

Figure 3.10 Probability density plot of simulated and estimated data for sequences of length $L = 400$. 86

Figure 3.11 Cumulative probability distribution plot of simulated and estimated data for sequences of length $L = 400$. 86

Chapter 1
Introduction: Where Mathematics Meets Biology



“Biology is mathematics’ next physics, only better”. Joel Cohen [12]

1.1. The Organized Complexity of Life

In 1953 Francis Crick, Rosalind Franklin, James Watson, and Maurice Wilkins determined the structure of DNA. This important event became a start of theoretical study of biological sequences. In his book “Life Itself. Its Origin and Nature” [14] Crick gave a very intuitive explanation of the general nature of life from an information-theoretical perspective, which can serve as a perfect start in establishing a connection between biology and mathematics.

He starts with the idea that, the high degree of *organized complexity* of life implies the need to store and process large amount of information. Thus, Nature has to provide an effective mechanism for this important task. Such a mechanism must perform two functions: information *storage* and *replication*. Information storage must be stable, which means that the information must be stored over a sufficiently long period of time, and replication must be accurate, which means that the copying can make only a small number of errors called *mutations*. Lack of accuracy can lead to accumulated errors and decay, but perfect accuracy is not required. Moreover, we must have a small number of mutations and require that they must be copied faithfully through generations. Most mutations lead to maladaptive changes but some might lead to improvement. There is no mechanism to produce only favorable mutations, which leaves chance to be the only source of true novelty. It is always possible for any rare chance event to become very common once this change is favored by the environment. This gives the living system its primary mechanism to improve itself. Note that neither *replication* nor *mutation* alone can explain both similarity and diversity of life, as we can see it everywhere around. Thus, the presence of both mechanisms is required [14].

Effectively dealing with large amounts of information in this manner puts some requirements on the mechanism. Recall that it needs to provide the system with easy storage, easy replication, easy error correction, and stability of information. The key idea is that an efficient way to do this requires the use of the combinatorial principle. That is, we express the information by using only a small number of types of standard units, but we combine them in many different ways. Similar to how we use words to form sentences in human language, life uses linear strings of standard units. It is required that in any language suitable for communication the number of words must be relatively small compared to the number of sentences they can produce. Nature follows the same principle requiring a small number of basic units, but the number of structures these units can form must be very large. Moreover, in order to carry information these structures must not be completely regular. The natural question is what Nature chose for these basic units and why.

Carbon is one the most abundant elements in the universe. It has a perfect balance between valence electrons available and valence electrons needed to fill the outer shell. This allows carbon to excel in bonding with itself and other atoms, and to form long chains, so it can build arbitrarily long and complex structures. Moreover, energy required to make and break carbon bonds is at the appropriate level for building molecules which are stable and reactive [46]. Note, that there are other elements that possess some of these properties but only carbon has all of them [47]. Taking all the above into account Nature's choice of carbon compounds as units is obvious. Crick concludes that information transmission with the required properties is naturally achieved through chemical reactions between combinatorial structures made of carbon compounds [14].

We will only consider one-dimensional linear structures, known as *biopolymers*, or biological sequences.

1.2. Biological Sequences and Their Patterns

In this section we consider sequences of three major macromolecules: DNA, RNA and proteins. In *Figure* 1.1 we show how each sequence participates in information transmission. This information flow model, known as *The Central Dogma of Molecular Biology*, was introduced by Francis Crick as early as in 1958 and published later in 1970. The main idea is that DNA replicates itself and through RNA codes for proteins.

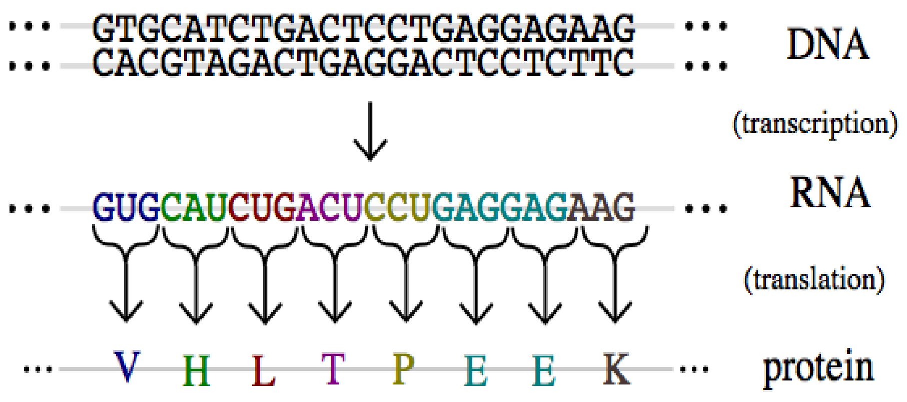


Figure 1.1. Central dogma of molecular biology [7].

Deoxyribonucleic acid or DNA is often called the basic material of life, because it carries genetic information enabling inheritance of properties from one living organism to another. The *bases* or *nucleotides* from which DNA sequences are combinatorially constructed consist of four carbon compounds: *Adenine* (A), *Guanine* (G), *Cytosine* (C), *Thymine* (T). The sequence made of these units is called a DNA *strand*. Each DNA molecule consists of two strands. The most important property of DNA sequences is strand complementarity, which is when pairs $A - T$ and $G - C$, joined to a phosphate-deoxyribose backbone, are linked together in a chain to form a DNA double helix. See *Figure* 1.2 and *Figure* 1.3.

One consequence of DNA complementarity is Chargaff's rule, which states that every DNA sample contains the same number of A's as T's, and the same number of G's as C's [11]. A further consequence is that it suffices to consider the bases in

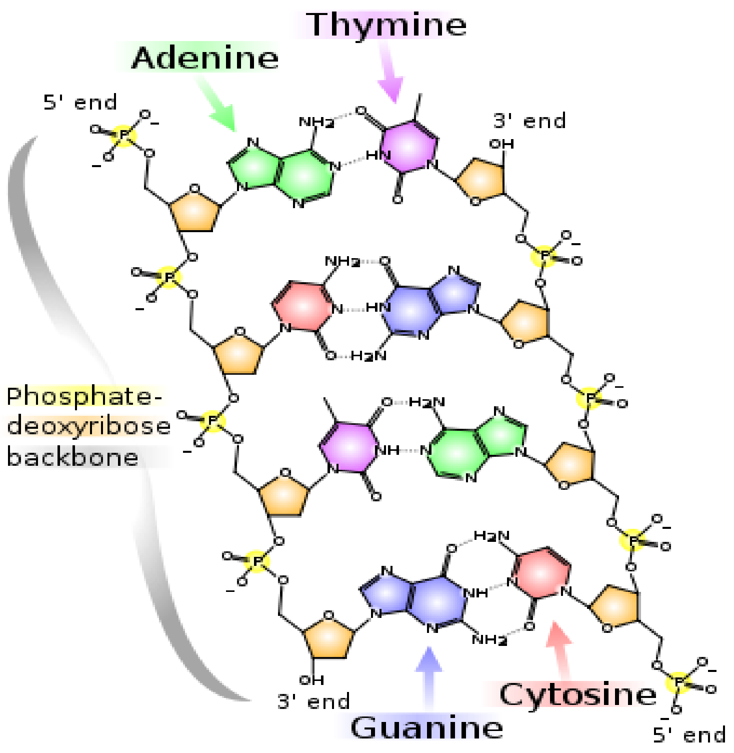


FIGURE 1.2. DNA chemical structure [6].

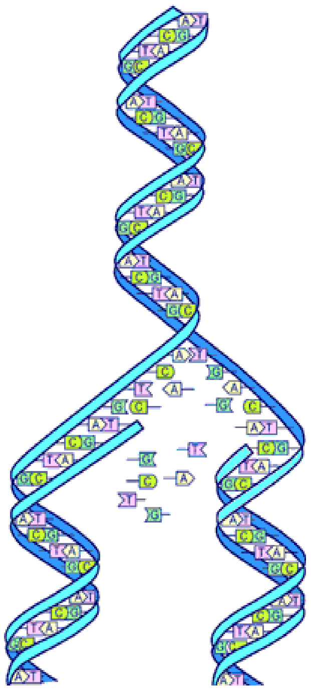


FIGURE 1.3. DNA complementarity [35].

only one strand of the double helix. It is important to note that each strand has a directionality that is determined by an asymmetrical arrangement of carbon atoms within the helix backbone. The asymmetry is indicated by writing the numbers $5'$ and $3'$ at the ends of the sequence. The convention is to write a single strand of DNA bases in the $5' \rightarrow 3'$ direction [36].

Ribonucleic acid or RNA is a biologically important type of molecule that consists of a long chain of nucleotide units. RNA is a single-stranded molecule. Its nucleotides contain ribose instead of deoxyribose in DNA (a type of ribose that lacks one oxygen atom). The alphabet of RNA sequence is very similar to that of DNA, with one exception: RNA has the base *Uracil* (U) rather than *Thymine* that is present in DNA. RNA is transcribed from DNA and then serves as an intermediary to protein synthesis. See *Figure* 1.1.

The third group of important molecules is called proteins. Their role is to control most of the chemical activity of the living body. Proteins (also known as polypeptides) are organic compounds made of amino acids arranged in a linear chain and folded into a globular form. The amino acids in a polymer are joined together by the peptide bonds between the carboxyl and amino groups of adjacent amino acid residues. The sequence of amino acids in a protein is translated from a *gene*, a major element of DNA and basic unit of heredity in a living organism. In general, the genetic code specifies 20 standard amino acids, which are often denoted by the 3-letter or 1-letter codes shown in *Table* 1.1. Proteins are essential parts of organisms and participate in virtually every process within cells.

All sequences must preserve some patterns in order to perform their function and interact with each other. For example, in *Figure* 1.4, a protein binds to DNA to prevent the transcription of certain genes.

The area of study that analyses patterns in biological sequences is called *biological sequence analysis*. Sequence analysis allows identification of genes and other conserved sequence patterns; they can be used to establish functional, structural, and

Amino Acid	**3-Letter**	**1-Letter**
Alanine	Ala	A
Arginine	Arg	R
Asparagine	Asn	N
Aspartic acid	Asp	D
Cycteine	Cys	C
Glutamic acid	Glu	E
Glutamine	Gln	Q
Glycine	Gly	G
Histidine	His	H
Isoleucine	Ile	I
Leucine	Leu	L
Lysine	Lys	K
Methionine	Met	M
Phenylalanine	Phe	F
Proline	Pro	P
Serine	Ser	S
Threonine	Thr	T
Tryptophan	Trp	W
Tyrosine	Tyr	Y
Valine	Val	V

Table 1.1. Amino acids and their abbreviations.

evolutionary relationship among proteins; and they provide a reliable method for inferring the biological functions of newly sequenced genes [10]. The basic idea is to be able to read any biological sentence, see and understand the genes and conserved patterns it encodes, as one would read a text in a familiar language.

For example, when a new protein is found, scientists usually have little idea about its function. Direct experimentation on a protein of interest is often costly and time-consuming [39]. As a result, one common approach to inferring the protein's function is to find similar proteins that have been studied and are stored in database. The functions of these proteins may give us a clue of the unknown function of our protein. If necessary, the information thus inferred can be verified experimentally.

One remarkable finding made through sequence comparison concerns the origins of cancer. In 1983, a paper by Doolittle, et al. [17] appearing in *Science* reported a 28-amino-acid sequence for platelet derived growth factors, a normal protein whose

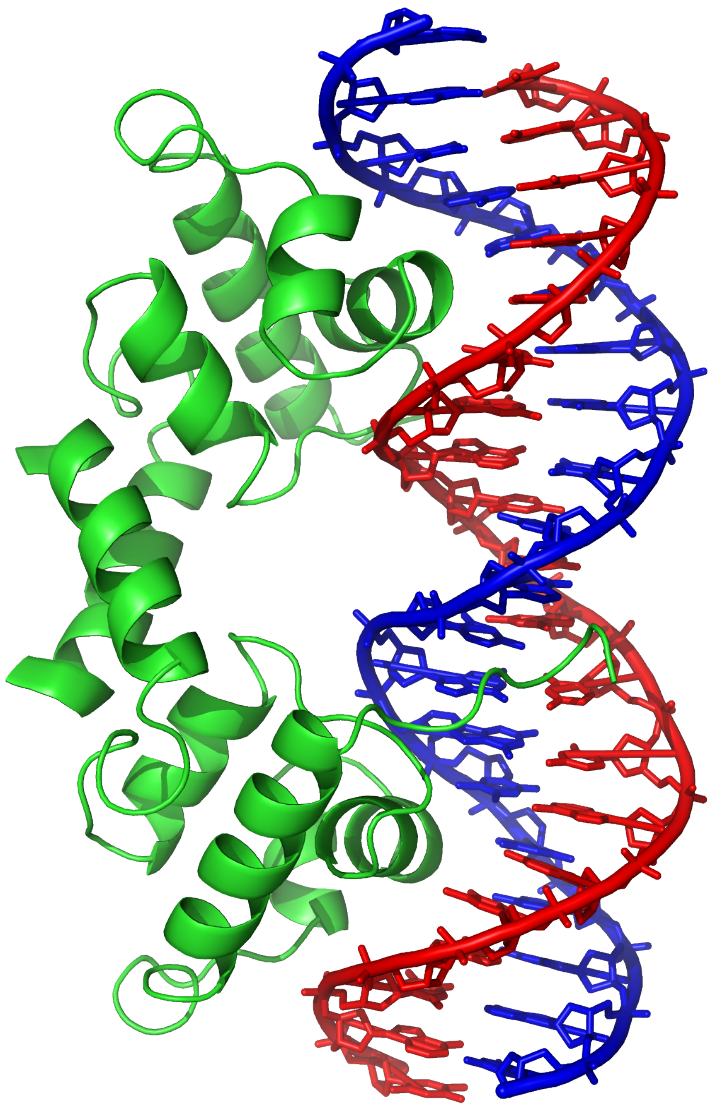


FIGURE 1.4. Lambda repressor protein bound to a lambda operator DNA sequence [45], [8].

function is to stimulate cell growth. By searching against a small protein database created by himself, Doolittle found that the sequence is almost identical to a sequence of ν-sis, an oncogene causing cancer in woolly monkeys. This finding changed the way oncogenesis is seen and understood. Today, it is generally accepted that cancer is caused by a normal growth gene which is switched on at the wrong time [10].

1.3. Biological Sequence Comparison

From now on we only look at pairs of biological sequences, where both sequences must represent either two DNA or two protein sequences. The major question we ask is how similar or *homologous* these pairs of sequences are.

Definition 1. We call two biological sequences *homologous* if they are evolutionarily similar, i.e., were derived from a common ancestor.

We want to be able to compare these sequences to see whether they are homologous or not. Then, the next step would be to find a method of comparison that measures the extent of this homology. The method that has been effectively used for both tasks is called *sequence alignment*.

The pairwise sequence alignment method compares a pair of sequences and assigns to it a value called a *score*. Three important aspects are involved in this process: computational, biological, and statistical.

From the computational perspective we can look at the alignment problem as the problem of comparing two text strings in a given alphabet. This problem has been extensively studied in the field of computer science. A purely computational solution to this problem will score a pair of text strings given an algorithm and a scoring rule. As a result, every pair of strings receives a score.

In the case of biological sequence comparison we want a *biologically relevant* alignment. By this we mean that we would like the resulting alignment to be probable, i.e., to have some chance of one sequence deriving from the other according to the explanation suggested by the alignment. This can be achieved by the special design of scoring rules, based on some essential molecular evolution assumptions combined with probabilistic techniques.

Finally, we need to understand what the fact that two sequences received a particular score can tell us about their homology. The field of mathematical statistics provides powerful tools to answer this question.

So far we showed some biological examples where the question of sequence comparison arises naturally and the answer to this question helps to explain some biological phenomena. The further presentation of material will go as follows: Chapter 2 takes a more detailed look at the sequence alignment problem from computational and biological perspectives. Chapter 3 is devoted to statistical methods that have been developed to assess the significance of alignment scores. At the end of Chapter 3 we also present and analyze the results of computer simulation that effectively illustrate the use of the computational, biological and statistical theory discussed in this work. We conclude by summarizing main results in Chapter 4.

Chapter 2

Sequence Alignment

“Share our similarities, celebrate our differences.” M. Scott Peck

2.1. Pairwise Sequence Alignment

A standard method for biological sequence comparison is sequence alignment. Sequence alignment is essentially nothing else but matching certain parts of two (or more) sequences. This, of course, can be done in many different way, so one must judge the goodness of such an alignment in some ways — preferably through a suitably defined model that is motivated by how evolution works on the molecular level. One then needs to find the best alignment according to our measure of goodness. To see how it is usually done in practice, we need to start with some basic definitions.

Definition 2. Let $\Sigma = \{\sigma_1, \ldots, \sigma_\ell\}$ be an alphabet, and let $X = x_1 x_2 \ldots x_m$ and $Y = y_1, y_2, \ldots y_n$ be two sequences over Σ. We extend our alphabet Σ by adding a special symbol "$-$", which denotes *space*. An *alignment* of sequences X and Y is a two-row matrix A with entries in $\Sigma \cup \{-\}$ such that

1. The first (second) row contains the letters of X (Y) in order;
2. One or more spaces "$-$" may appear between two consecutive letters of Σ in each row;
3. Each column contains at least one letter of Σ [10].

From a biological point of view, an alignment of two sequences poses a hypothesis about how the sequences evolved from their most recent common ancestor. An alignment accounts for the following three mutational events:

- Substitution (also called point mutation) - A nucleotide (amino acid) is replaced by another.
- Insertion - One or several nucleotides (amino acids) are inserted at a position in a sequence.
- Deletion - One of several nucleotides (amino acids) are deleted from a sequence [10].

Definition 3. For $1 \leqslant i, j \leqslant \ell$ and $x_i, y_j \in \Sigma$ a column $\binom{x_i}{y_j}$ of an alignment A is called a *match* if $x_i = y_j$ and *mismatch* (or *substitution*) if $x_i \neq y_j$. A column $\binom{-}{y_j}$ is called an *insertion* and a column $\binom{x_i}{-}$ is called a *deletion*. Columns $\binom{-}{y_j}$ and $\binom{x_i}{-}$ are also called *indels*. The column $\binom{-}{-}$ is not allowed in an alignment. Also, it is common to refer to columns of the alignment as *sites*.

Definition 4. A *gap* in an alignment is defined as a sequence of spaces "$-$" located between two letters in a row. We will use p to denote a gap length. We say an alignment is *gapped* or *ungapped* to emphasize whether indels are allowed or not in the alignment.

For example, in the alignment

$$\mathcal{A} = \begin{pmatrix} A & - & T & A & C & - & T & G & G \\ - & G & T & C & C & G & T & - & G \end{pmatrix}$$

the two sequences are identical in the third, fifth, seventh, and ninth columns - they are the matches. There is a mismatch in the fourth column, first and eighth columns are deletions, and second and sixth columns are insertions.

We need a rule to tell us the "goodness" of each possible alignment. Traditionally, a scoring matrix is used for this purpose.

Definition 5. Consider an alphabet $\Sigma = \{\sigma_1, \ldots, \sigma_\ell\}$ and a function $s : \Sigma \times \Sigma \to \mathbb{Z}$ that assigns a *score* $s(\sigma_i, \sigma_j)$ to each pair of letters (σ_i, σ_j). Then for $1 \leqslant i, j \leqslant \ell$ a matrix $(s(\sigma_i, \sigma_j))$ is called a *scoring matrix* for an alphabet Σ.

In the case of gapped alignment, having a scoring matrix is not enough. We also need a way to score gaps.

Definition 6. A *gap penalty function* is a function $\Delta : k \to \mathbb{Z}^- \cup \{0\}$ that assigns a non-positive integer penalty $\Delta(k)$ for every gap of length $1 \leqslant k \leqslant \max(n, m)$ in an alignment.

Definition 7. A scoring matrix and a gap penalty function are called a *scoring scheme.*

Definition 8. An *alignment score* is the sum of scores of each pair of symbols in the alignment. This important property is called *additivity.*

2.2. Scoring Schemes

2.2.1. Biological Perspective. The major biological question we want to answer is whether the two sequences are homologous; thus, we would like to have such a scoring scheme that answers this question as correctly as possible. When sequence comparison was first introduced, the scoring schemes were extremely simple. For example, for each match the score of "1" was used and for each mismatch and insertion or deletion a value of " -1" were assigned. This rule was too simple to produce a biologically meaningful result. Then, biologists who became experienced enough with sequence data started using their knowledge to design more sophisticated scoring rules. All of these rules were somewhat unsubstantiated and usually worked just for a very small number of problems.

It is intuitively obvious that taking random sequences that were generated according to a given underlying letter-distribution, and aligning these sequences randomly, the frequency of letter pairs should differ depending on the distribution of the letters. Even in these chance alignments, pairs of frequent letters appear more often than pairs of unfrequent letters. In real alignments, which have biological meaning, one would expect identities and conservative substitutions to be more frequent than they would appear in chance alignments, thus, these columns should contribute positive terms to the score of an alignment. Similarly, non-conservative changes should be less frequent in real alignments than they would be in chance alignments, therefore the columns corresponding to non-conservative changes should contribute negative terms to the alignment score. In the absence of a rigorous theory, experience helped scientists to decide whether an aligning pair is conservative or not and up to what degree, thus allowing them to come up with scores that "work" based on intuition. Later, developments in computer technology and the increase in sequence data resources led to much more advanced and rigorous scoring scheme design based on probability theory and statistical data analysis.

2.2.2. Events and Their Probabilities.

In everyday life we often make statements about a chance of some event A happening, where A can be an event of having a rain tomorrow or of obtaining a sequence alignment score of k or higher. The occurrence or non-occurrence of A can be a result of a chain of circumstances. Such circumstances are called *experiments* or *trials*. The result of an experiment is called its *outcome*.

Definition 9. A set of all possible outcomes of an experiment is called the *sample space* and is denoted by Ω.

Definition 10. A nonempty collection $\mathcal{F}$ of subsets of Ω is called a *σ-field* if it satisfies the following conditions:

(a) $\varnothing \in \mathcal{F}$;
(b) if $A_1, A_2, \ldots \in \mathcal{F}$ then $\bigcup_{i=1}^{\infty} A_i \in \mathcal{F}$;
(c) if $A \in \mathcal{F}$ then $A^c \in \mathcal{F}$.

Thus, we may associate a $\{\Omega, \mathcal{F}\}$ with any experiment, where Ω is the set of all possible outcomes or *elementary events* and $\mathcal{F}$ is a σ-field of subsets of Ω which contains all the events we may be interested.

Definition 11. A *probability measure* $\mathbb{P}$ on $(\Omega, \mathcal{F})$ is a function $\mathbb{P} : \mathcal{F} \to [0,1]$ satisfying

(a) $\mathbb{P}[\varnothing] = 0, \mathbb{P}[\Omega] = 1$;
(b) if $A_1, A_2, \ldots$ is a collection of disjoint members of $\mathcal{F}$, in that $A_i \cap A_j = \varnothing$ for all pairs i, j satisfying $i \neq j$, then

$$\mathbb{P}\left[\bigcup_{i=1}^{\infty} A_i\right] = \sum_{i=1}^{\infty} \mathbb{P}[A_i].$$

Definition 12. The triple $(\Omega, \mathcal{F}, \mathbb{P})$ is called a *probability space.*

Lemma 2.1. *A probability measure* $\mathbb{P}$ *has following properties:*

(a) $\mathbb{P}[A^c] = 1 - \mathbb{P}[A]$,

(b) *if* $B \supseteq A$ *then* $\mathbb{P}[B] = \mathbb{P}[A] + \mathbb{P}[B \setminus A] \geq \mathbb{P}[A]$,

(c) *if* $A_1, A_2, \ldots, A_n$ *are events, then*

$$\mathbb{P}\left[\bigcup_{i=1}^{n} A_i\right] = \sum_i \mathbb{P}[A_i] - \sum_{i<j} \mathbb{P}[A_i \cap A_j] + \sum_{i<j<k} \mathbb{P}[A_i \cap A_j \cap A_k] - \ldots$$

$$+ (-1)^{n+1}\mathbb{P}[A_1 \cap A_2 \cap \ldots \cap A_n] = \sum_{k=1}^{n} (-1)^{n-1} \sum_{\substack{J : J \subseteq \{1,2,\ldots,n\} \\ |J|=k}} \mathbb{P}\left(\bigcap_{j \in J} A_j\right).$$

Parts (a) and (b) follow directly from Definition 11, part (b), and part (c) of the lemma can be proved by induction on n, using Definition 11 and set-properties.

Definition 13. If $\mathbb{P}[B] > 0$ then the *conditional probability* that A occurs given that B occurs is defined to be

$$\mathbb{P}[A|B] = \frac{\mathbb{P}[A \cap B]}{\mathbb{P}[B]}.$$

Definition 14. Events A and B are called *independent* if

$$\mathbb{P}[A \cap B] = \mathbb{P}[A]\mathbb{P}[B].$$

More generally, a family $\{A_i : i \in I\}$ is called *independent* if

$$\mathbb{P}\left[\bigcap_{i \in J} A_i\right] = \prod_{i \in J} \mathbb{P}[A_i]$$

for al subsets J of I.

2.2.3. Substitution Matrices. In this section we discuss the probabilistic model that allowed the construction of modern substitution matrices used for aligning sequences with no gaps allowed. Therefore the two sequences that are aligned have the same length.

Given a pair of aligned sequences, we want to assign a score to an alignment that gives a measure of the relative likelihood that the sequences are related as opposed to being unrelated. We do this by having models that assign a probability to the alignment in each of the two cases; we then consider the ratio of the two probabilities.

Recall that we want our scoring scheme to be additive. This suggests the assumption that mutations at different sites of the sequence occur independently, treating a gap of arbitrary length as a single mutation. Although we know that interactions between residues play a critical role in determining protein structure, the assumption of independence appears to be a reasonable approximation for DNA and protein sequences. Still, although it is possible to take these dependencies into account, doing so gives rise to significant computational complexities.

Traditionally, such complexities are avoided by considering the *random* model R. In the random model, each letter a has a frequency q_a, and at any position of the sequence the probability that this letter occurs is q_a, independently of what happened at other positions. Moreover, we assume that the alignment occured as a result of aligning two such random sequences of the same length. Hence the probability of seeing this alignment in the random model is just the product of the probabilities of seeing these two sequences. Because of the independence of the positions, the probability of each sequence is just the product of the frequencies of the amino acids appearing in the sequence. Thus, the probability of the alignment in the random model is:

$$\mathbb{P}[x, y|R] = \prod_i q_{x_i} \prod_j q_{y_j}. \tag{2.2.1}$$

Thus, in the random model the probability that at any given site we have an aligned residue-pair where the first residue is a and the second residue is b is $q_a q_b$. Note that for $a \neq b$ we have that the probability of observing a and b together in an alignment (regardless of order) is $2q_a q_b$, and the probability that a appears with itself in an alignment is q_a^2. The frequency in which a appears in the alignment is therefore

$$\frac{1}{2}\left(2q_a^2 + 2\sum_{a \neq b} q_a q_b\right) = q_a \sum_b q_b = q_a,$$

as expected.

In the alternative *match* model M a pair of aligned residues where the first residue is a and the second residue is b has a frequency p_{ab}, and these aligned residues appear at any site of the alignment with their given frequency, independently of the other sites. The value p_{ab} can be thought of as the probability that the residues a and b each have been derived independently from some unknown original residue c in their common ancestor, here c might be the same as a and/or b. While theoretically we do not need $p_{ab} = p_{ba}$, there is no real difference between the two sequences, thus in practice we have $p_{ab} = p_{ba}$, and for $a \neq b$ the probability that a, b appears in the same alignment regardless of the order is $2p_{ab}$. This gives a probability for the whole alignment of

$$\mathbb{P}[x, y|M] = \prod p_{x_i y_i}.$$

The ratio of these two likelihoods is known as the *odds* ratio:

$$\frac{\mathbb{P}[x, y|M]}{\mathbb{P}[x, y|R]} = \frac{\prod_i p_{x_i y_i}}{\prod_i q_{x_i} \prod_i q_{y_i}} = \prod_i \frac{p_{x_i y_i}}{q_{x_i} q_{y_i}}.$$

In order to arrive at an additive scoring system, we take the logarithm of this ratio, known as the *log-odds ratio*:

$$R = \sum_i r(x_i, y_i) \tag{2.2.2}$$

where

$$r(a, b) = \log\left(\frac{p_{ab}}{q_a q_b}\right) \tag{2.2.3}$$

is the log-likelihood ratio of the residue pair (a, b) occurring as an aligned pair, as opposed to an unaligned pair.

An additional scaling parameter λ is also often used to make all the entries in the matrix integers

$$s(a, b) = \frac{1}{\lambda} r(a, b) = \frac{1}{\lambda} \log\left(\frac{p_{ab}}{q_a q_b}\right). \tag{2.2.4}$$

This gives us

$$S = \sum_i s(x_i, y_i). \tag{2.2.5}$$

The scaling parameter's purpose is purely algorithmic — the computer deals with integer arithmetic easier than with floating point arithmetic.

As we wanted, equation (2.2.5) is a sum of individual scores $s(a, b)$ for each aligned pair of residues. Note, that if we expect to see a pair (a, b) aligned in homologous sequences more often then they might occur by chance, then $p_{ab} > q_a q_b$, leading to a positive score. In this way conservative substitutions receive positive scores and nonconservative substitutions receive negative scores. The $s(a, b)$ scores can be arranged in a matrix.

In order to estimate target frequencies p_{ab} we first need to count the frequencies of each pair in a dataset of *trusted* pairwise alignments. This set of alignments must be chosen from a database. The more specific information is available about a pair of sequences we want to align, the better we can choose the set of alignments for target frequencies estimation. For example, in integral membrane proteins hydrophobic residues appear more often, so if we estimate the target frequencies using only sequences of integral domain proteins it will be biased towards hydrophobicity.

Another important parameter that must be taken into account is the *evolutionary distance* between two sequences. The evolutionary distance measures how recently two sequences diverged from each other. The target frequencies must favor identical residues in case of a recent divergence of two sequences. This bias should decrease with an increase of divergence. Thus, usually a series of related scoring matrices is developed, each to reflect an appropriate degree of divergence.

For proteins, for instance, this gives rise to a 20×20 matrix $M = (s(a_i, a_j))$, with $s(a_i, a_j)$ in position i, j in the matrix, where a_i, a_j are the ith and jth amino acid in some ordering. This is known as a *score matrix* or a *substitution matrix*. A family of matrices called $BLOSUM$ ($BLOcks\ SUbstitution\ Matrix$) is currently one of the most widely used family of protein substitution matrices.

In BLOSUM matrices the target frequencies were estimated from the database of *protein blocks* [27]. Steven and Jorja Henikoff designed an automated system for finding and assembling the most highly conserved regions called *blocks* of related proteins. Their method could determine the best set of non-overlapping blocks for any database of protein sequences [26]. The Henikoffs then derived substitution matrices from about 2000 blocks of aligned sequence segments characterizing more then 500 groups of related proteins [27].

Let us take a closer look at the procedure followed by the Henikoffs as it is described in their paper [27]. Consider a single block representing a conserved region of a protein family. For a new member of this family, we seek a set of scores for matches and mismatches that most favors a correct alignment with each of the other segments in the block relative to an incorrect alignment. For each column of the block, we first count the number of matches and mismatches of each type between the new sequence and every other sequence in the block.

For example, if the residue of the new sequence that aligns with the first column of the first block is A and the column has 9 A residues and 1 S residue, then there are 9 AA matches and 1 AS mismatch. This procedure is repeated for all columns of all blocks with the summed results stored in a table. The new sequence is added to the group. For another new sequence, the same procedure is followed, summing these numbers with those already in the table. Notice that successive addition of each sequence to the group leads to a table consisting of counts of all possible amino acid pairs in a column. For example, in the column consisting of 9 A residues and 1 S residue, there are $8+7+\ldots+1 = 36$ possible AA pairs, 9 AS or SA pairs, and no SS pairs. Counts of all possible pairs in each column of each block in the database are summed. So, if a block has a width of n amino acids and a depth of l sequences, it contributes $\frac{nl(l-1)}{2}$ amino acid pairs to the count: $\frac{(1\times 10\times 9)}{2} = 45$ in the above example. The result of this counting is a frequency table listing the number of times each of the $20 + 19 + \ldots + 1 = 210$ different amino acid pairs occur among the blocks. The table is used to calculate a matrix representing the odds ratio between these observed frequencies and those expected by chance.

Let the total number of amino acid i, j pairs ($1 \leq j \leq i \leq 20$) for each entry of the frequency table be denoted f_{ij}. Then the observed probability of occurrence of each i, j pair is

$$p_{ij} = \frac{f_{ij}}{\sum_{i=1}^{20} \sum_{j=1}^{i} f_{ij}}.$$

For the column of 9 A residues and 1 S residue in the example, where $f_{AA} = 36$ and $f_{AS} = 9$, $p_{AA} = \frac{36}{45} = 0.8$ and $p_{AS} = \frac{9}{45} = 0.2$. Next, we estimate the expected probability of occurrence for each i, j pair. It is assumed that the observed pair frequencies are those of the population. For example, 36 pairs have A in both positions of the pair and 9 pairs have A at only one of the two positions, so that the expected probability of A in a pair is $\frac{(36+(9/2))}{45} = 0.9$ and that of S is $\frac{9/2}{45} = 0.1$. In general, the probability of occurrence for the ith amino acid in an i, j pair is

$$q_i = p_{ii} + \sum_{j \neq i} p_{ij}/2.$$

The expected probability of occurrence for each i, j pair is then $e_{ij} = q_i q_j$ for $i = j$ and $e_{ij} = q_i q_j + q_j q_i$ for $i \neq j$. In the example, the expected probability of AA is $0.9 \times 0.9 = 0.81$, that of $AS + SA$ is $2 \times (0.9 \times 0.1) = 0.18$, and that of SS is $0.1 \times 0.1 = 0.01$. An odds ratio matrix is calculated where each entry is $\frac{p_{ij}}{e_{ij}}$. A log-odds ratio is then calculated in bit units as

$$r(i, j) = \log_2 \left(\frac{q_{ij}}{e_{ij}} \right).$$

If the observed frequencies are as expected, $r(i, j) = 0$; if less then expected $r(i, j) < 0$; if more then expected, $r(i, j) > 0$. Log-odds ratios are multiplied by a scaling factor $\lambda = 2$ and then rounded to the nearest integer value to produce $s(i, j)$ entrees for

BLOSUM matrices in half-bit units, where

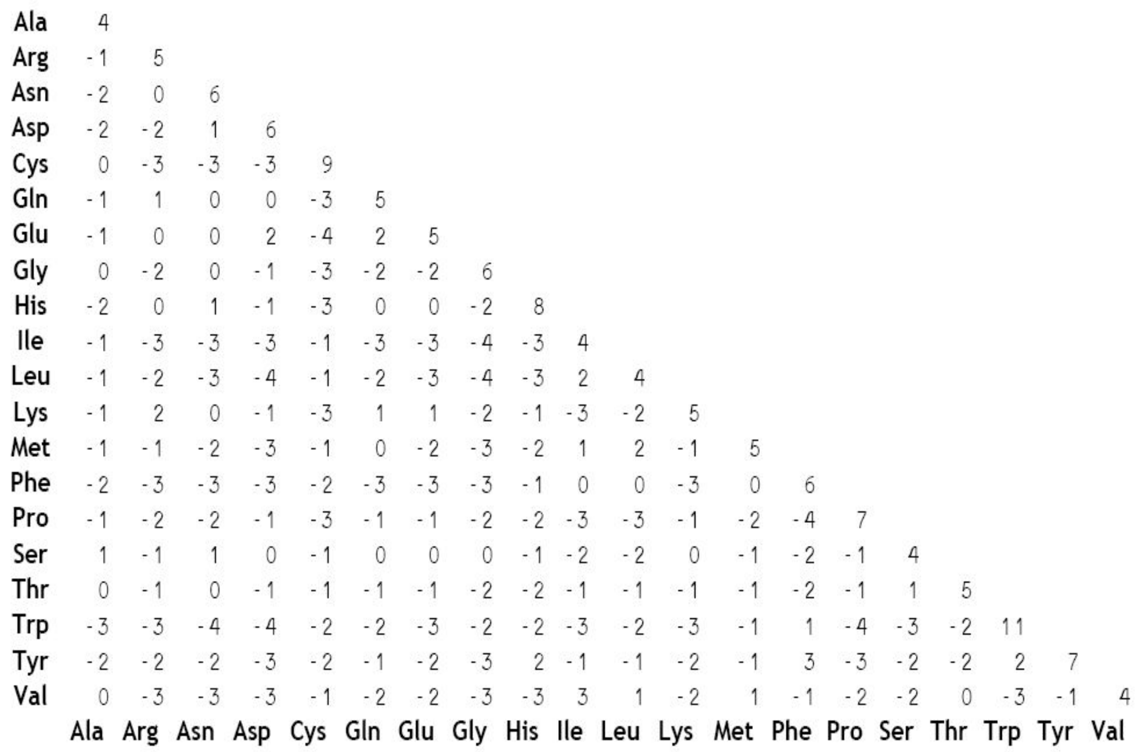
$$s(i,j) = \frac{1}{2} r(i,j) = \frac{1}{2} \log_2\left(\frac{q_{ij}}{e_{ij}}\right).$$

For each substitution matrix the average mutual information per amino acid pair H (also called relative entropy), and the expected score E in bit units is given by

$$H = \sum_{i=1}^{20} \sum_{j=1}^{i} p_{ij} s(i,j);$$

$$E = \sum_{i=1}^{20} \sum_{j=1}^{i} q_i q_j s(i,j).$$

Matrices with comparable relative entropies also have similar expected scores.



	Ala	Arg	Asn	Asp	Cys	Gln	Glu	Gly	His	Ile	Leu	Lys	Met	Phe	Pro	Ser	Thr	Trp	Tyr	Val
Ala	4																			
Arg	-1	5																		
Asn	-2	0	6																	
Asp	-2	-2	1	6																
Cys	0	-3	-3	-3	9															
Gln	-1	1	0	0	-3	5														
Glu	-1	0	0	2	-4	2	5													
Gly	0	-2	0	-1	-3	-2	-2	6												
His	-2	0	1	-1	-3	0	0	-2	8											
Ile	-1	-3	-3	-3	-1	-3	-3	-4	-3	4										
Leu	-1	-2	-3	-4	-1	-2	-3	-4	-3	2	4									
Lys	-1	2	0	-1	-3	1	1	-2	-1	-3	-2	5								
Met	-1	-1	-2	-3	-1	0	-2	-3	-2	1	2	-1	5							
Phe	-2	-3	-3	-3	-2	-3	-3	-3	-1	0	0	-3	0	6						
Pro	-1	-2	-2	-1	-3	-1	-1	-2	-2	-3	-3	-1	-2	-4	7					
Ser	1	-1	1	0	-1	0	0	0	-1	-2	-2	0	-1	-2	-1	4				
Thr	0	-1	0	-1	-1	-1	-1	-2	-2	-1	-1	-1	-1	-2	-1	1	5			
Trp	-3	-3	-4	-4	-2	-2	-3	-2	-2	-3	-2	-3	-1	1	-4	-3	-2	11		
Tyr	-2	-2	-2	-3	-2	-1	-2	-3	2	-1	-1	-2	-1	3	-3	-2	-2	2	7	
Val	0	-3	-3	-3	-1	-2	-2	-3	-3	3	1	-2	1	-1	-2	-2	0	-3	-1	4

FIGURE 2.1. BLOSUM62 substitution matrix [24].

The evolutionary distance was taken into account in BLOSUM matrices by clustering segments within blocks. This is done by specifying a clustering percentage in which sequence segments that are identical for at least that percentage of amino acids are grouped together. In this way, varying the clustering percentage leads to a family of BLOSUM matrices: BLOSUM80, BLOSUM70, BLOSUM62, and BLOSUM45. For example, to construct BLOSUM62 percentage of 62% is used. Clustering at 62% reduces the number of blocks contributing to the table by 25% with the remainder contributing 1.25 million pairs (including fractional pairs), whereas without clustering, $>$ 15 million pairs are counted.

Henikoffs were able to show that BLOSUM matrices perform better then other general purpose substitution matrices. They chose a group of protein sequences called the *guanine nucleotide-binding protein-coupled receptors.* This particularly challenging group had been previously used to test searching and alignment programs. From 114 family members three distinct sequences were chosen for queries. Henikoffs performed these queries against the *PROSITE* 8.0 database, a database of protein families and domains [28], using BLOSUM and other matrices. They showed that when performing search using BLOSUM matrices more sequences from the target group could be detected [27]. They also conducted similar study for other 504 protein groups stored in *PROSITE* 8.0 database and showed that BLOSUM62 performed slightly better then other BLOSUM matrices [27].

It is also important to keep in mind that the scoring is based on log-odds ratios, not just the frequencies of observed aligned pairs. For example, in a substitution matrix BLOSUM62, if we compare identity scores for tryptophan (W) and leucine (L) pairs, we see that they are considerably different $s(W,W) = 11$, and $s(L,L) = 4$. At first glance, this might seem counterintuitive, since in the BLOSUM62 training dataset the estimated target frequencies for leucine and tryptophan were $p_{LL} = 0.0371$ and $p_{WW} = 0.0065$. However, statistically tryptophan is a much rarer amino acid then leucine, more precisely, $q_W = 0.013$ and $q_L = 0.099$. So, we get

$$s(W,W) = \frac{1}{0.347} \ln\left(\frac{0.0065}{0.013^2}\right) = 10.5,$$

and

$$s(L,L) = \frac{1}{0.347}\ln\left(\frac{0.0371}{0.099^2}\right) = 3.8,$$

where $\lambda = 0.347$ is the scaling parameter. In the BLOSUM62 matrix these scores are rounded to $s(W,W) = 11$, and $s(L,L) = 4$ [19].

2.2.4. Gap Penalties. There is no general theory available for guiding the choice of gap costs. The scoring of gaps were usually chosen with two things in mind: one, that the resulting algorithms that compute the alignments and scores are effective, the other, that the resulting alignments are reasonable.

The most straightforward scheme is to charge a fixed penalty for each indel, i.e.,

$$\Delta_{CONST}(k) = dk$$

for a gap of length k, $d = \mathit{const}$. However, biologically, a gap of length $k > 1$ is more likely to happen than k gaps of length 1. This is because a single mutational event usually results in insertion or deletion of a stretch of characters. Moreover, separated gaps are due to distinct mutational events and by the previous argument tend to be longer. Over the years, it has been observed that the optimal alignments produced by the constant gap scheme usually contain a large number of short gaps and are often not biologically meaningful.

To capture the idea that a single mutational event might insert or delete a sequence of residues, Waterman and Smith [41] introduced the *affine gap penalty model*. Under this model, the penalty

$$\Delta_{AFF}(k) = o + ek$$

is charged for a gap of length $k \geq 1$, where o is a large penalty for opening a gap and e a smaller penalty for extending it.

The affine gap cost is based on the hypothesis that gap length has an exponential distribution, that is, the probability of a gap of length $k \geq 1$ is $\alpha(1-\beta)\beta^k$ for some constants α and β. Under this hypothesis, an affine gap cost is derived by charging

$\log(\alpha(1-\beta)\beta^k)$ for a gap of length k. But this hypothesis might not be true in general. For instance the study of Benner, Cohen, and Gonnet [9] suggests that the frequency of a length k is accurately discribed by $mk^{-1.7}$ for some constant m, i.e., a power law.

A *generalized* affine gap cost is introduced by Altschul [1]. A generalized gap consists of a consecutive sequence of indels in which spaces can be in either row. A generalized gap of length 10 may contain 10 insertions; it may also contain 4 insertions and 6 deletions. To reflect the structural property of a generalized gap, a generalized affine gap cost has three parameters a, b, c. The score $-a$ is introduced for the opening of a gap; $-b$ is for each residue inserted or deleted; and $-c$ is for each pair of residues left unaligned. For a gap with $k \geq 0$ insertions and $l \geq 0$ deletions such that $k+\ell \geq 1$ a generalized gap penalty is

$$\Delta_{GEN}(k,\ell) = -(a + |k-\ell|b) + c\min\{k,\ell\}.$$

Generalized affine gap costs can be used for locally or globally aligning protein sequences. The empirical study of Zachariah et al. [49] shows that this generalized affine gap cost model improves significantly the accuracy of protein alignment.

Sometimes it is more convenient to think of an alignment in the following way

$$\begin{array}{ccccccccc} \mathbf{D} & \mathbf{I} & \mathbf{H} & \mathbf{H} & \mathbf{H} & \mathbf{I} & \mathbf{H} & \mathbf{D} & \mathbf{H} \end{array}$$
$$\mathcal{A} = \begin{pmatrix} A & - & T & A & C & - & T & G & G \\ - & G & T & C & C & G & T & - & G \end{pmatrix}.$$

The additional line of letters $\{H, D, I\}$, where H, I, D stand for homology, insertion, and deletion, denotes each column by its type. With this representation in mind it is more convenient to organize gap penalties in a matrix. Consider a 3×3 matrix

$$W = \begin{pmatrix} w(I,I) & w(I,H) & w(I,D) \\ w(H,I) & w(H,H) & w(H,D) \\ w(D,I) & w(D,H) & w(D,D) \end{pmatrix},$$

where $w(x, y)$, for $x, y \in \{H, I, D\}$ denotes a penalty that is added to the alignment score when we move from one column to the next in the alignment, considering only the fact whether each column represents homology, deletion, or insertion. For example, $w(H, D)$ denotes the penalty of going from a homology column of the alignment to a deletion column [36].

For example, in the case of a constant penalty $\Delta_{CONST}(k) = dk$,

$$W_{CONST}(d) = \begin{pmatrix} d & 0 & d \\ d & 0 & d \\ d & 0 & d \end{pmatrix}.$$

Or, in the case of affine penalty $\Delta_{AFF}(k) = o + ek$,

$$W_{AFF}(o, e) = \begin{pmatrix} e & 0 & e \\ o+e & 0 & o+e \\ e & 0 & e \end{pmatrix}.$$

These matrices will be useful in the next section.

2.3. Alignment Algorithms

2.3.1. Computational Perspective. Once we set a proper scoring scheme, deciding that two biological sequences are similar is no different from deciding that two text strings are similar. One set of methods for biological sequence analysis is therefore rooted in computer science, where there is an extensive literature on string comparison methods [18].

Definition 15. The alignment graph $G_{n,m}$ is the directed graph on the set of nodes $\{0, 1, \dots, n\} \times \{0, 1, \dots, m\}$ and three classes of directed edges as follows:

- (diagonal) edges labeled by S between pairs of nodes $(i, j) \to (i + 1, j + 1)$, where $0 \le i \le n - 1$ and $0 \le j \le m - 1$;
- (horizontal) edges labeled by I between pairs of nodes $(i, j) \to (i, j + 1)$, where $0 \le i \le n$ and $0 \le j \le m - 1$;
- (vertical) edges labeled by D between pairs of nodes $(i, j) \to (i+1, j)$, where $0 \le i \le n - 1$ and $0 \le j \le m$ [36].

An alignment graph together with vertex $(0, 0)$ as a source and vertex (n, m) as a sink forms an alignment network.

All possible alignments correspond one-to-one to the directed paths from the source to the sink in the alignment network. Thus, the alignment network gives a compact representation of all possible alignments of the sequences. The alignments of two sequences of length n and m and the directed paths from source to sink in the alignment graph $G_{n,m}$ are in one-to-one correspondence with each other. A diagonal edge of the form $(i - 1, j - 1) \to (i, j)$ corresponds to a site where the i-th letter of the first sequence is aligned with the j-th letter of the second sequence, and horizontal/vertical edges of the form $(i - 1, j - 1) \to (i - 1, j)$ and $(i - 1, j - 1) \to (i, j - 1)$ correspond to a $-$ being aligned with the j-th letter of the second sequence or the i-th letter of the first sequence, respectively. Thus, the alignment network gives a compact representation of all possible alignments of the two sequences. *Figure* 2.2

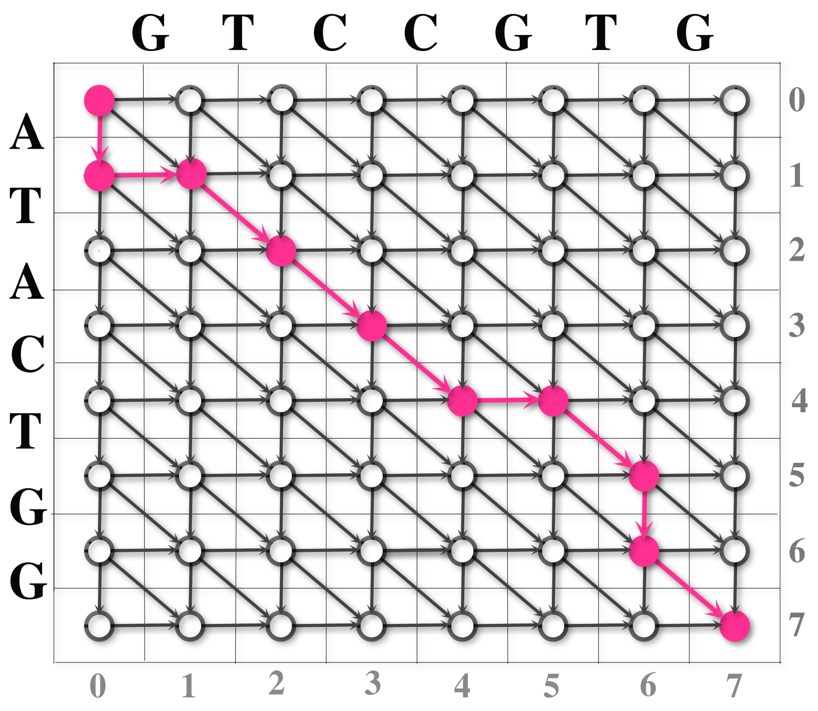


FIGURE 2.2. Aligment graph for sequences ATACTGG and GTC-CGTG [10].

shows an example from [10], where $\Sigma = \{A, C, G, T\}$, $n = m = 7$, $X = ATACTGG$, $Y = GTCCGTG$, and the highlighted path in the alignment network is

$$P_{\mathcal{A}} = (0,0) \to (1,0) \to (1,1) \to (2,2) \to (3,3) \to (4,4) \to (4,5) \to (5,6) \to (6,6) \\ \to (7,7).$$

This path represents the following alignment

$$\mathcal{A} = \begin{pmatrix} A & - & T & A & C & - & T & G & G \\ - & G & T & C & C & G & T & - & G \end{pmatrix}$$

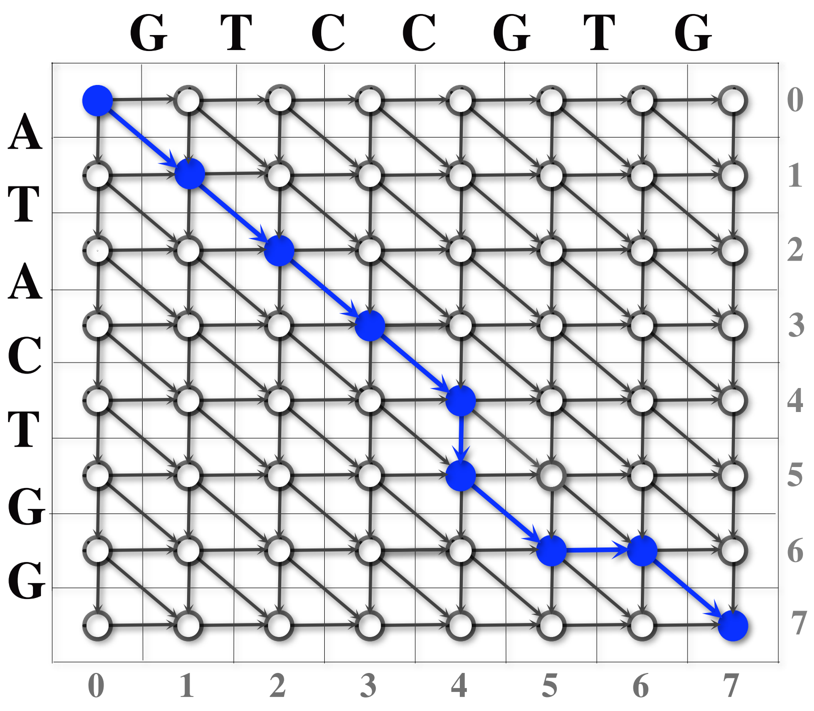


FIGURE 2.3. Another alignment graph for sequences ATACTGG and GTCCGTG [10].

On the other hand, we can align X and Y in the following way

$$\mathcal{A}' = \begin{pmatrix} A & T & A & C & T & G & - & G \\ G & T & C & C & - & G & T & G \end{pmatrix}$$

As shown in *Figure* 2.3 this alignment will be represented by the following path in the alignment network

$$P_{\mathcal{A}'} = (0,0) \to (1,1) \to (2,2) \to (3,3) \to (4,4) \to (5,4) \to (6,5) \to (6,6) \to (7,7).$$

Theorem 2.2. *An alignment of two sequences corresponds uniquely to a path from the source to the sink in the alignment network of these two sequences. In this correspondence, matches and mismatches are represented by diagonal edges, and insertions and deletions by vertical and horizontal edges respectively.*

The aim of alignment is to find the best one of all the possible alignments of two sequences. Hence, a natural question is to ask is: How many possible alignments are there for two sequences of length n and m? By the previous Theorem 2.2 this is equivalent to asking how many different paths there are from the source $(0, 0)$ to the sink (n, m) in an alignment network [10].

Definition 16. Given complex-valued functions f and g of natural number variable n we write

$$f \approx g \text{ as } n \to \infty$$

to express the fact that

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 1.$$

and f and g are called *asymptotically equivalent* as $n \to \infty$.

Theorem 2.3. *Let $a(n, m)$ be the number of possible alignments of two sequences with lengths n and m. Then for all $n, m \geq 1$ and $k \geq 0$ the following holds*

$$a(n, m) = a(n - 1, m) + a(n - 1, m - 1) + a(n, m - 1)$$

$$a(0, k) = a(k, 0) = 1,$$

and

$$a(n, m) = \sum_{k=\max(n,m)}^{n+m} \binom{k}{n} \binom{n}{n + m - k}.$$

In particular, when $m = n$ we have

$$a(n, n) \geq \binom{2n}{n} \approx \frac{2^{2n}}{\sqrt{\pi n}}.$$

Proof. In an alignment network the paths from the source $(0,0)$ to a vertex (i,j) correspond to possible alignments of two sequences of length i and j. Hence, $a(i,j)$ is equal to the number of paths from $(0,0)$ to the vertex (i,j). Because every path from the source $(0,0)$ to the sink (m,n) must go through exactly one of the vertices $(m-1,n)$, $(m,n-1)$, and $(n-1,m-1)$, we obtain $a(n,m) = a(n-1,m) + a(n-1,m-1) + a(n,m-1)$.

There is only one way to align the empty sequence to a non-empty sequence. Thus for any $k \geq 0$ we have $a(0,k) = a(k,0) = 1$.

Suppose there are k columns in the alignment with n letters and $k-n$ spaces in the first row and m letters and $k-m$ spaces the second row. This means that we must have $\min(n,m) \leq k$, since the number of columns in the alignment is at least as much as the length of the longer sequence, and $k \leq m+n$, since for more than $m+n$ columns we should have two spaces aligned, which is not allowed.There are $\binom{k}{n}$ possible ways to arrange the n letters in the first row. For each such arrangement $k-m$ spaces in the second row must be placed below the letters in the first row giving $\binom{n}{k-m} = \binom{n}{n+m-k}$ possible arrangements. Thus we have $a_k(n,m) = \binom{k}{n}\binom{n}{n+m-k}$ possible k-column alignments. Since $\max(n,m) \leq k \leq m+n$ the number of all possible alignments of two sequences with lengths n and m can be calculated by the formula

$$a(n,m) = \sum_{k=\max(n,m)}^{n+m} \binom{k}{n}\binom{n}{n+m-k}.$$

In case $n = m$ we have

$$a(n,n) = \sum_{k=n}^{2n} \binom{k}{n}\binom{n}{2n-k} = \sum_{k=n}^{2n-1} \binom{k}{n}\binom{n}{2n-k} + \binom{2n}{n} \geq \binom{2n}{n}.$$

Applying Stirling's approximation

$$x! \approx \sqrt{2\pi}x^{x+\frac{1}{2}}e^{-x},$$

we obtain

$$a(n,n) \geq \binom{2n}{n} \approx \frac{2^{2n}}{\sqrt{\pi n}}$$

as desired [10]. □

For example consider two sequences of length 100. By Theorem 2.3 we have $a(100, 100) > 2^{196} > 10^{59}$, an astronomically large number. Clearly, this shows that it is definitely not feasible to examine all possible alignments and that we need an efficient method for sequence alignment.

To take gap penalties into account we generalize the notion of an alignment graph to an *extended* alignment graph which is a graph obtained by replacing each vertex in the alignment graph with a *directed complete bipartite* graph on 6 vertices with edges weighted by corresponding values of a gap penalty matrix W. This procedure is shown in detail in *Figure* 2.4.

An extended alignment graph weighted according to a scoring scheme with vertices $(0, 0)$ as a source and (n, m) as a sink forms an *extended* alignment network.

With an appropriate scoring scheme used to weigh the corresponding edges of an alignment network, finding an alignment with maximum score, i.e., an optimal alignment is equivalent to finding a directed path of maximum length in the extended alignment network.

2.3.2. Dynamic Programming. Recall that, while searching for an optimal alignment, we can not simply compute all possible alignments, due to the fact that their number is quite large. Thus, there is a need for an efficient computational method. One such method is known as *dynamic programming*.

The dynamic-programming paradigm applies when subproblems overlap — that is, when subproblems share subproblems. A dynamic-programming algorithm solves each subproblem just once and then saves its answer in a table, thereby avoiding the work of recomputing the answer every time it solves each subproblem.

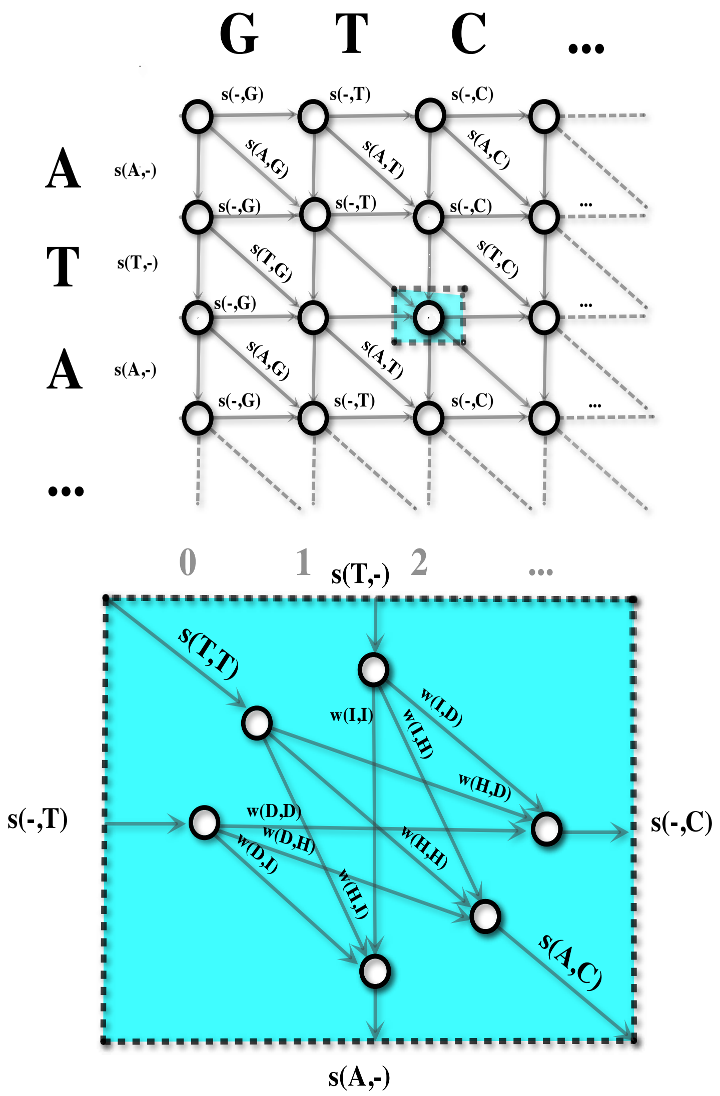


FIGURE 2.4. The extended alignment graph, obtained by replacing each vertex of the alignment graph with a directed complete bipartite graph on 6 vertices weighted by corresponding values of the gap penalty matrix W [36].

We typically apply dynamic programing to *optimization problems*. Such problems can have many possible solutions. Each solution has a value, and we wish to find a solution with the optimal (minimum or maximum) value. We call such solution *an* optimal solution to the problem, as opposed to *the* optimal solution, since there may be several solutions that achieve the same optimal value.

When developing a dynamic-programming algorithm, we follow a sequence of four steps:

1. Characterize the structure of an optimal solution.
2. Recursively define the value of an optimal solution.
3. Compute the value of an optimal solution, typically in a botton-up fashion.
4. Construct an optimal solution from the computed information.

Steps $1-3$ form the basis of a dynamic-programming solution to a problem. If we need only the value of an optimal solution, and not the solution itself, then we can omit step 4. When we do perform step 4, we sometimes maintain additional information during step 3 so that we can easily construct an optimal solution [13].

In the case of optimal sequence alignment a dynamic-programming algorithm in general follows a sequence of three steps:

1. An initialization of dynamic-programming matrix (or matrices) for remembering optimal scores of subproblems.
2. A recursive definition of the optimal score and a bottom-up approach of filling the matrix by solving the smallest subproblems first.
3. A backtracking procedure to recover the structure of the optimal solution that gave the optimal score. This procedure is often called a *traceback* in standard literature [13].

2.3.3. Global Pairwise Alignment. We first consider a problem of the optimal global alignment between two sequences with gaps. The *Needleman-Wunsch* algorithm [34] is a standard dynamic-programming algorithm for solving this problem. A more efficient version was later introduced by Gotoh [22].

We want to build an optimal gapped alignment with constant gap penalty $\delta(k) = dk$ using previous solutions for optimal alignments of smaller subsequences. Suppose we are aligning two sequences $X = x_1 \dots x_n$ and $Y = y_1 \dots y_m$, S is an $(n+1)\times(m+1)$ matrix indexed by (i, j), and $S(i, j)$ is the score of the best possible alignment between initial segments $x_1 \dots x_i$ and $y_1 \dots y_j$. We can define a recursion for building $S(i, j)$.

We also need two $(n+1)\times(m+1)$ matrices T_1, T_2 to store the backtracking indices, used in the traceback step. As before, $s(x_i, y_j)$ is the score of aligning a pair (x_i, y_j).

1. Initialization

   The initial step is $S(0,0) = 0$. Then we proceed to fill the matrix from top left to bottom right. There are some boundary conditions we have to take in account. Along the top row, where $j = 0$, the values $S(i, j-1)$ and $S(i-1, j-1)$ are not defined, so the values $S(i, 0)$ must be handled specially. The values $S(i, 0)$ represent alignments of a prefix of x to a sequence of all gaps in y, so we define $S(i, 0) = -id$ and $S(0, j) = -jd$ by analogy for $1 \leq i \leq n$, and $1 \leq j \leq m$. Also, $T_1(0,0) = T_1(i,0) = T_1(0,j) = 0$, and $T_2(0,0) = T_2(i,0) = T_2(0,j) = 0$, for $1 \leq i \leq n$, and $1 \leq j \leq m$.

2. Recursion

   If we already know $S(i-1, j-1), S(i-1, j)$, and $S(i, j-1)$, we can calculate $S(i,j)$ as follows:

$$S(i,j) = \max \begin{cases} S(i-1, j-1) + s(x_i, y_j), \\ S(i-1, j) - d, \\ S(i, j-1) - d. \end{cases} \tag{2.3.1}$$

$$T_1(i,j) = \begin{cases} i-1, \text{ if } S(i,j) = S(i-1, j-1) + s(x_i, y_j) \\ \qquad \text{or } S(i,j) = S(i-1, j) - d; \\ i, \text{ if } S(i,j) = S(i, j-1) - d. \end{cases} \tag{2.3.2}$$

$$T_2(i,j) = \begin{cases} j-1, \text{ if } S(i,j) = S(i-1, j-1) + s(x_i, y_j) \\ \qquad \text{or } S(i,j) = S(i, j-1) - d; \\ j, \text{ if } S(i,j) = S(i-1, j) - d. \end{cases} \tag{2.3.3}$$

We fill in matrices S, T_1, and T_2 by repeatedly applying formulas (2.3.1), (2.3.2), and (2.3.3) respectively at each step (i, j).

3. Traceback

By definition, $S(n, m)$ is the best score for a global alignment of sequences X and Y. To find an alignment itself we must traceback a path of choices from (2.3.1) that led to the final value. We start from the final cell and follow the indices we stored in T_1 and T_2 when building the matrix S. At each step in the traceback process, we move back from the current cell (i, j) to the cell $(T_1(i, j), T_2(i, j))$, which must be the one of the cells $(i-1, j-1)$, $(i-1, j)$ or $(i, j-1)$ from which the value of $S(i, j)$ was derived. We also prepend a pair of symbols onto the front of the current alignment: a substitution pair $\binom{x_i}{y_j}$ if the step was to $(i-1, j-1)$, a deletion pair $\binom{x_i}{-}$ if the step was to $(i-1, j)$, and an insertion pair $\binom{-}{y_j}$ if the step was to $(i, j-1)$. At the end we will reach the start of the matrix, $i = j = 0$ [18].

2.3.4. Local Pairwise Alignment. More often, we seek the best alignment between subsequences of $X = x_1 \dots x_n$ and $Y = y_1 \dots y_m$. This is usually a more sensitive way to detect similarity when comparing very highly divergent sequences. This happens because often it is only a part of a sequence that is preserved with substantial similarity over time, while the rest of the sequence can accumulate too many mutations. The *Smith-Waterman* algorithm [41] is a standard algorithm for finding the best alignment between subsequences of X and Y.

Similar to the global case, we are aligning two sequences $X = x_1 \dots x_n$ and $Y = y_1 \dots y_m$ in a gapped alignment with constant gap penalty $\delta(k) = dk$. $\hat{S}$ is an $(n+1) \times (m+1)$ matrix indexed by (i, j), and $\hat{S}(i, j)$ is the score of the best possible alignment between subsequences of initial segments $x_1 \dots x_i$ and $y_1 \dots y_j$. We can define a recursion for building $\hat{S}(i, j)$. Again, we need two $(n+1) \times (m+1)$ matrices T_1, T_2 to store the backtracking indices, used in the traceback step.

Since this algorithm is looking for the highest scoring alignment of a subsequence, we might as well try to restart the alignment of the two sequences once the score computed goes below 0, since we can achieve higher score that way. This results in two major differences in the algorithm. First, we will set the score $\hat{S}(i,j)$ to 0 when $S(i,j)$ would become negative in the Needleman-Wunsch algorithm. This will lead to the start of a new alignment, thus the backtracking procedure must end at one of the cells (i,j), such that $\hat{S}(i,j) = 0$. Second, we need to maintain the current highest alignment score M and its position $(i^\star, j^\star)$ in $\hat{S}$.

1. Initialization

$$\hat{S}(0,0) = \hat{S}(i,0) = \hat{S}(0,j) = 0, \text{ for } 1 \le i \le n, 1 \le j \le m.$$

$$T_1(0,0) = T_1(i,0) = T_1(0,j) = 0,$$

and

$$T_2(0,0) = T_2(i,0) = T_2(0,j) = 0, \text{ for } 1 \le i \le n, 1 \le j \le m.$$

$$M = 0, \text{ and } i^\star = j^\star = 0.$$

2. Recursion

$$\hat{S}(i,j) = \max \begin{cases} 0, \\ \hat{S}(i-1,j-1) + s(x_i, y_j), \\ \hat{S}(i-1,j) - d, \\ \hat{S}(i,j-1) - d. \end{cases} \tag{2.3.4}$$

$$T_1(i,j) = \begin{cases} i-1, \text{ if } \hat{S}(i,j) = \hat{S}(i-1,j-1) + s(x_i, y_j) \\ \qquad \text{or } \hat{S}(i,j) = \hat{S}(i-1,j) - d; \\ i, \text{ if } \hat{S}(i,j) = \hat{S}(i,j-1) - d \\ \qquad \text{or } \hat{S}(i,j) = 0. \end{cases} \tag{2.3.5}$$

$$T_2(i,j) = \begin{cases} j-1, \text{ if } \hat{S}(i,j) = \hat{S}(i-1,j-1) + s(x_i, y_j) \\ \qquad \text{or } \hat{S}(i,j) = \hat{S}(i,j-1) - d; \\ j, \text{ if } \hat{S}(i,j) = \hat{S}(i-1,j) - d \\ \qquad \text{or } \hat{S}(i,j) = 0. \end{cases} \tag{2.3.6}$$

$$(i^\star, j^\star) = \begin{cases} (i,j), \text{ if } \hat{S}(i,j) > M; \\ \text{remains unchanged otherwise} . \end{cases} \tag{2.3.7}$$

$$M = \begin{cases} \hat{S}(i,j), \text{ if } \hat{S}(i,j) > M; \\ \text{remains unchanged otherwise} . \end{cases} \tag{2.3.8}$$

3. Traceback

Now alignment can end anywhere in the matrix, so instead of taking the value in the bottom right corner, $\hat{S}(n,m)$, for the best possible score, we look for $M = \hat{S}(i^\star, j^\star)$, which is the highest value of $\hat{S}(i,j)$ over the whole matrix. We start at the cell $(i^\star, j^\star)$ in $\hat{S}$. At each step in the traceback process, we move back from the current cell (i,j) to the cell $(T_1(i,j), T_2(i,j))$, which must be the one of the cells $(i-1,j-1)$, $(i-1,j)$, $(i,j-1)$, or (i,j) from which the value of $\hat{S}(i,j)$ was derived. As in the global case, we prepend a pair of symbols onto the front of the current alignment: a substitution pair $\binom{x_i}{y_j}$ if the step was to $(i-1,j-1)$, a deletion pair $\binom{x_i}{-}$ if the step was to $(i-1,j)$, and an insertion pair $\binom{-}{y_j}$ if the step was to $(i,j-1)$. The traceback ends if we move from (i, j) to (i, j), i.e. if we remain at the same position. This means that we have reached a point where $\hat{S}(i, j) = 0$, and our local alignment started from here [18].

If the expected score of a random match $\sum_{a,b} p_a p_b s(a, b)$ is nonnegative, then in any long random, unrelated pair of sequences we should still find long, highscoring local alignments just by chance. Thus, in a biologically meaningful scoring sequence the expected score of a random match must be negative. Also, to be able to obtain a local alignment, we must have at least one positive score $s(a, b)$.

2.3.5. Pairwise Alignment with Affine Gap Penalty. To adapt the above dynamic-programming algorithm to the case of affine gap penalty, $\Delta_{AFF}(k) = o + ek$, we need to introduce two additional $(n+1) \times (m+1)$ matrices D and I, where $D(i, j)$ represents the score of the best possible alignment between initial segments $x_1 \dots x_i$ and $y_1 \dots y_j$, given that x_i is aligned to a gap, and $I(i, j)$ represents the score of the best possible alignment between initial segments $x_1 \dots x_i$ and $y_1 \dots y_j$, given that y_j is aligned to a gap. Now, $S(i, j)$ represents the score of the best possible alignment between initial segments $x_1 \dots x_i$ and $y_1 \dots y_j$, given that x_i is aligned to y_j. Also, $T_1(0, 0) = T_1(i, 0) = T_1(0, j) = 0$, and $T_2(0, 0) = T_2(i, 0) = T_2(0, j) = 0$, for $1 \leq i \leq n$, and $1 \leq j \leq m$.

For global alignment we have the following algorithm:

1. Initialization

$$S(0, 0) = 0,\ S(i, 0) = S(0, j) = -\infty,\ \text{for } i, j \geq 1,$$

$$D(i, 0) = o + ei,\ D(0, j) = -\infty,$$

$$I(0, j) = o + ej,\ I(i, 0) = -\infty;$$

Also,

$$T_1(0, 0) = T_1(i, 0) = T_1(0, j) = 0,$$

and

$$T_2(0,0) = T_2(i,0) = T_2(0,j) = 0, \text{ for } 1 \le i \le n, 1 \le j \le m.$$

2. Recursion

$$S(i,j) = s(x_i, y_j) + \max \begin{cases} S(i-1, j-1), \\ D(i-1, j-1), \\ I(i-1, j-1), \end{cases} \tag{2.3.9}$$

$$D(i,j) = \max \begin{cases} S(i-1, j) + o + e, \\ D(i-1, j) + e, \end{cases} \tag{2.3.10}$$

$$I(i,j) = \max \begin{cases} S(i, j-1) + o + e, \\ I(i, j-1) + e; \end{cases} \tag{2.3.11}$$

Let $S_{\max}(i,j) = \max\{S(i,j), D(i,j), I(i,j)\}$, then

$$T_1(i,j) = \begin{cases} i-1, \text{ if } S_{\max}(i,j) = S(i,j) \\ \qquad \text{or } S_{\max}(i,j) = D(i,j); \\ i, \text{ if } S_{\max}(i,j) = I(i,j). \end{cases} \tag{2.3.12}$$

$$T_2(i,j) = \begin{cases} j-1, \text{ if } S_{\max}(i,j) = S(i,j) \\ \qquad \text{or } S_{\max}(i,j) = I(i,j); \\ j, \text{ if } S_{\max}(i,j) = D(i,j). \end{cases} \tag{2.3.13}$$

3. Traceback

We start at the cell (n,m). At each step in the traceback process, we move back from the current cell (i,j) to the cell $(T_1(i,j), T_2(i,j))$, which must be the one of the cells $(i-1, j-1)$, $(i-1, j)$, or $(i, j-1)$. We also prepend a pair of symbols onto the front of the current alignment: a substitution pair

$\binom{x_i}{y_j}$ if the step was to $(i-1, j-1)$, a deletion pair $\binom{x_i}{-}$ if the step was to $(i-1, j)$, and an insertion pair $\binom{-}{y_j}$ if the step was to $(i, j-1)$, and stop once we reach the cell $(0, 0)$.

Similar to the case with a constant gap, in local alignment with affine gap penalty, we look for the highest scoring alignment between subsequences of $X = x_1 \ldots x_n$ and $Y = y_1 \ldots y_m$. Thus, we set $\hat{S}(i, j)$ to 0, i.e. restart the alignment, each time the Needleman-Wunsch score $S(i, j)$ becomes negative. We also maintain the current highest alignment score M and its position $(i^\star, j^\star)$ in $\hat{S}$. The major difference as opposed to the affine gapped global case, is that in the case of local alignment we must start and end the alignment with a matching character. Thus, the algorithm goes as follows:

1. Initialization

$$\hat{S}(0,0) = \hat{S}(i,0) = \hat{S}(0,j) = 0, \text{ for } i, j \geq 1,$$

$$D(i,0) = D(0,j) = I(0,j) = I(i,0) = -\infty;$$

$$T_1(0,0) = T_1(i,0) = T_1(0,j) = 0,$$

and

$$T_2(0,0) = T_2(i,0) = T_2(0,j) = 0, \text{ for } 1 \leq i \leq n, 1 \leq j \leq m.$$

$$M = 0, \text{ and } i^\star = j^\star = 0.$$

2. Recursion

$$\hat{S}(i,j) = \max \begin{cases} 0, \\ \hat{S}(i-1,j-1) + s(x_i,y_j), \\ D(i-1,j-1) + s(x_i,y_j), \\ I(i-1,j-1) + s(x_i,y_j), \end{cases} \tag{2.3.14}$$

$$D(i,j) = \max \begin{cases} \hat{S}(i-1,j) + o + e, \\ D(i-1,j) + e, \end{cases} \tag{2.3.15}$$

$$I(i,j) = \max \begin{cases} \hat{S}(i,j-1) + o + e, \\ I(i,j-1) + e; \end{cases} \tag{2.3.16}$$

Let $\hat{S}_{\max}(i,j) = \max\{\hat{S}(i,j), D(i,j), I(i,j)\}$, then

$$T_1(i,j) = \begin{cases} i-1, \text{ if } \hat{S}_{\max}(i,j) = \hat{S}(i,j) \\ \qquad \text{or } \hat{S}_{\max}(i,j) = D(i,j); \\ i, \text{ if } \hat{S}_{\max}(i,j) = I(i,j) \\ \qquad \text{or } \hat{S}(i,j) = 0. \end{cases} \tag{2.3.17}$$

$$T_2(i,j) = \begin{cases} j-1, \text{ if } \hat{S}_{\max}(i,j) = \hat{S}(i,j) \\ \qquad \text{or } \hat{S}_{\max}(i,j) = I(i,j); \\ j, \text{ if } \hat{S}_{\max}(i,j) = D(i,j) \\ \qquad \text{or } \hat{S}(i,j) = 0. \end{cases} \tag{2.3.18}$$

$$(i^\star, j^\star) = \begin{cases} (i,j), \text{ if } \hat{S}(i,j) > M; \\ \text{remains unchanged otherwise .} \end{cases} \tag{2.3.19}$$

$$M = \begin{cases} \hat{S}(i,j), \text{ if } \hat{S}(i,j) > M; \\ \text{remains unchanged otherwise .} \end{cases} \tag{2.3.20}$$

3. Traceback

   As in the constant penalty case, we first look for $M = \hat{S}(i^\star, j^\star)$, which is the highest value of $\hat{S}(i,j)$ over the whole matrix. We start at the cell $(i^\star, j^\star)$ in $\hat{S}$. At each step in the traceback process, we move back from the current cell (i,j) to the cell $(T_1(i,j), T_2(i,j))$, which must be the one of the cells $(i-1, j-1)$, $(i-1, j)$, $(i, j-1)$, or (i,j). As in the global case, we prepend a pair of symbols onto the front of the current alignment: a substitution pair $\binom{x_i}{y_j}$ if the step was to $(i-1, j-1)$, a deletion pair $\binom{x_i}{-}$ if the step was to $(i-1, j)$, and an insertion pair $\binom{-}{y_j}$ if the step was to $(i, j-1)$. The traceback ends if the step was to (i,j), which means that we have reached a value $\hat{S}(i,j) = 0$ [18].

2.3.6. Examples of Global and Local Alignments. We now provide examples of global and local alignments with gaps of two DNA sequences taken directly from the paper [37]. We align two subsequences

$$X[0, 10] = TACTAGCGCA$$

and

$$Y[0, 10] = ACGGTAGATT$$

of sequences drawn from the nucleotide alphabet $\Sigma = \{A, C, G, T\}$. We use the standard nucleotide scoring scheme, with $s(a, b) = 5$ if $a = b$ and -4 otherwise, and the affine gap penalty $\Delta(k) = -3 - 2k$, where k is the length of the gap. The corresponding scoring matrix $(s(\sigma_i, \sigma_j))$, $\sigma_i, \sigma_j \in \Sigma$, $1 \leq i, j \leq 4$ is

$$(s(\sigma_i,\sigma_j)) = \begin{pmatrix} s(A,A) & s(A,C) & s(A,G) & s(A,T) \\ s(C,A) & s(C,C) & s(C,G) & s(C,T) \\ s(G,A) & s(G,C) & s(G,G) & s(G,T) \\ s(T,A) & s(T,C) & s(T,G) & s(T,T) \end{pmatrix} = \begin{pmatrix} 5 & -4 & -4 & -4 \\ -4 & 5 & -4 & -4 \\ -4 & -4 & 5 & -4 \\ -4 & -4 & -4 & 5 \end{pmatrix}.$$

The corresponding gap penalty matrix W is

$$W_{AFF}(-3,-2) = \begin{pmatrix} -2 & 0 & -2 \\ -5 & 0 & -5 \\ -2 & 0 & -2 \end{pmatrix}.$$

A fragment of the extended alignment network corresponding to this scoring scheme is shown in *Figure* 2.5.

The optimal global path shown in *Figure* 2.6 ends at the point $(10, 8)$, e.g., consists of 12 edges, in the following order:

$$P_{\mathcal{A}} = (0,0) \rightarrow (0,1) \rightarrow (2,1) \rightarrow (3,2) \rightarrow (3,3) \rightarrow (3,4) \rightarrow (4,5) \rightarrow (5,6) \rightarrow (6,7)$$
$$\rightarrow (7,7) \rightarrow (8,7) \rightarrow (9,8) \rightarrow (10,8).$$

It corresponds to the global sequence alignment of $X[0,10]$ and $Y[0,8]$,

$$\mathcal{A} = \begin{pmatrix} T & A & C & - & - & T & A & G & C & G & C & A \\ - & A & C & G & G & T & A & G & - & - & - & A \end{pmatrix}$$

In *Figure* 2.7 the cell (i, j) displays the global score $S(i, j)$, calculated from equation (2.3.1).

The global score $S(10, 8) = -5+5+5-7+5+5+5-9+5 = 9$ is the sum of the corresponding edges and represents the path of greatest weight starting at $(0, 0)$ and

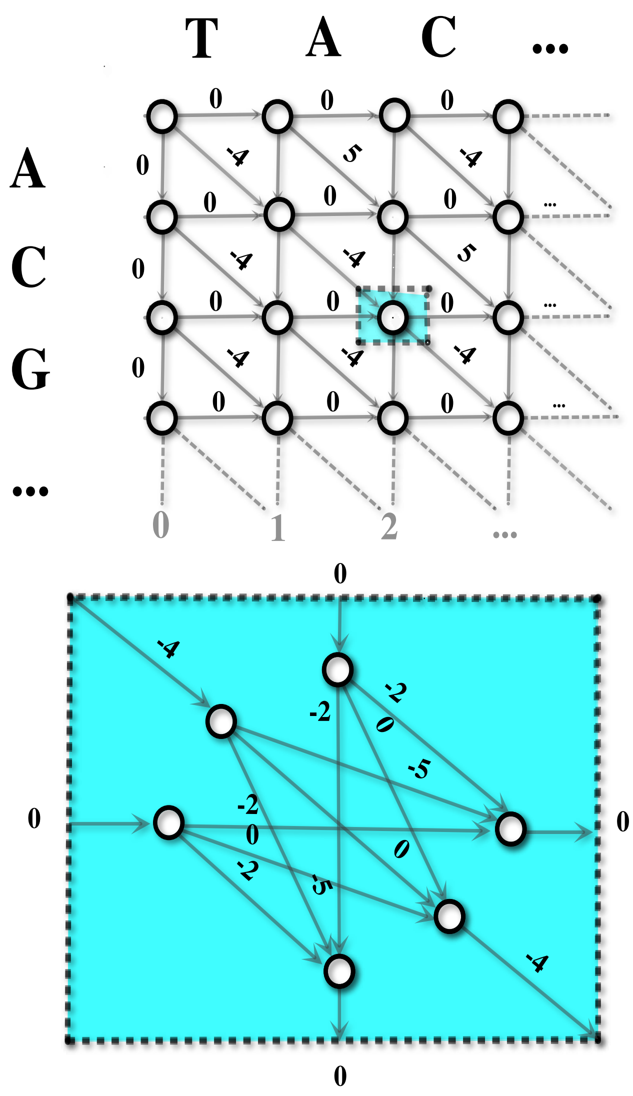


FIGURE 2.5. A fragment of an extended alignment network of sequences $X[0,10] = TACTAGCGCA$ and $Y[0,10] = ACGGTAGATT$.

ending at $(10,8)$. Note that the global maximum for X and Y can be no less than 13, the largest global score shown.

The optimal local path shown in *Figure* 2.8 ends at the point $(6,7)$, e.g., consists of 7 edges, in the following order:

$$P_{\hat{\mathcal{A}}} = (1,0) \to (2,1) \to (3,2) \to (3,3) \to (3,4) \to (4,5) \to (5,6) \to (6,7).$$

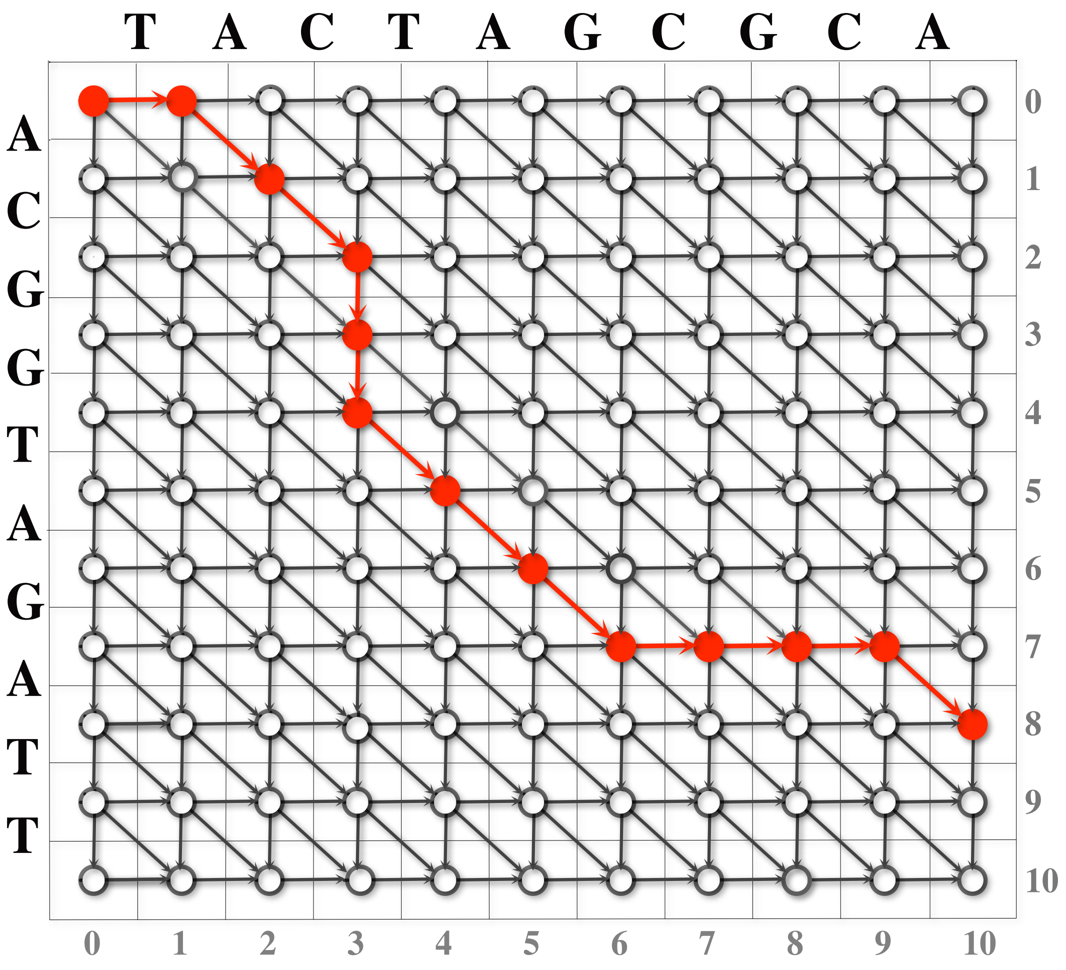
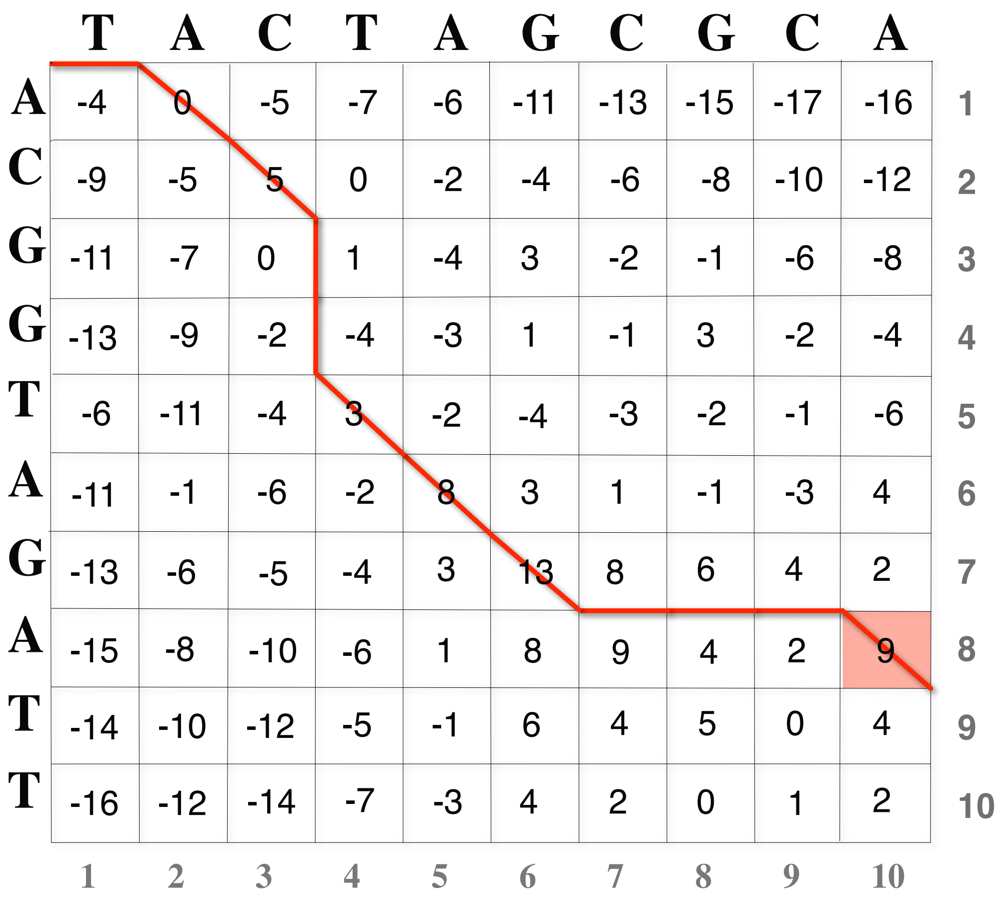


FIGURE 2.6. The optimal global path in the alignment network.

It corresponds to the local sequence alignment of, e.g., $X[0,10]$, and $Y[0,10]$,

$$\hat{\mathcal{A}} = \begin{pmatrix} A & C & - & - & T & A & G \\ A & C & G & G & T & A & G \end{pmatrix}$$

In *Figure* 2.9 each cell (i,j) displays the corresponding local score $\hat{S}(i,j)$, which can be calculated from the recursion (2.3.4). A local score of 0 indicates that no path of positive weight ends at the corresponding point. The local score $\hat{S}(6,7) = 5+5-7+5+5+5 = 18$ is the sum of the corresponding edges and represents the path of greatest weight ending at $(6,7)$. The score is also the greatest weight of



	T	A	C	T	A	G	C	G	C	A	
A	-4	0	-5	-7	-6	-11	-13	-15	-17	-16	1
C	-9	-5	5	0	-2	-4	-6	-8	-10	-12	2
G	-11	-7	0	1	-4	3	-2	-1	-6	-8	3
G	-13	-9	-2	-4	-3	1	-1	3	-2	-4	4
T	-6	-11	-4	3	-2	-4	-3	-2	-1	-6	5
A	-11	-1	-6	-2	8	3	1	-1	-3	4	6
G	-13	-6	-5	-4	3	13	8	6	4	2	7
A	-15	-8	-10	-6	1	8	9	4	2	9	8
T	-14	-10	-12	-5	-1	6	4	5	0	4	9
T	-16	-12	-14	-7	-3	4	2	0	1	2	10
	1	2	3	4	5	6	7	8	9	10	

FIGURE 2.7. Global alignment matrix ($S(i,j)$).

any path within the rectangle displayed, so it also corresponds to the local maximum score $\hat{M}(10,10) = 18$.

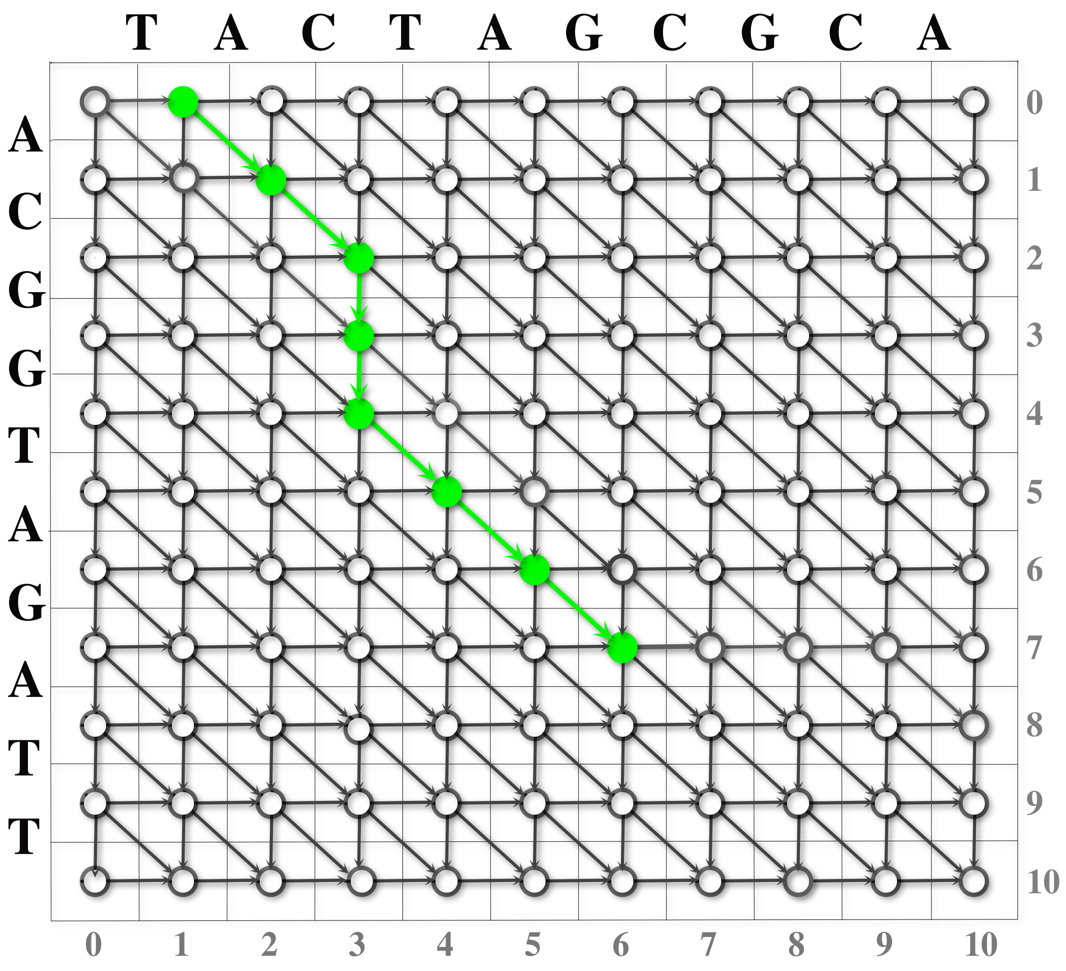
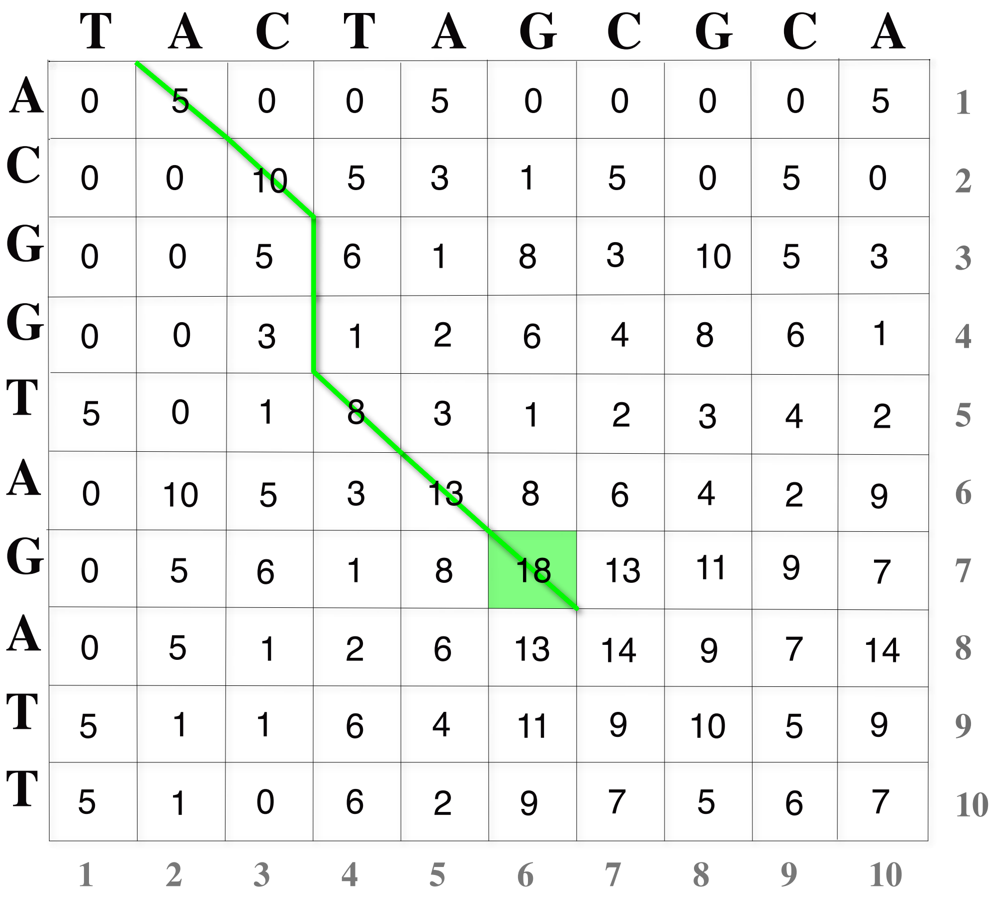


Figure 2.8. The optimal local path in the alignment network.



	T	A	C	T	A	G	C	G	C	A	
A	0	5	0	0	5	0	0	0	0	5	1
C	0	0	10	5	3	1	5	0	5	0	2
G	0	0	5	6	1	8	3	10	5	3	3
G	0	0	3	1	2	6	4	8	6	1	4
T	5	0	1	8	3	1	2	3	4	2	5
A	0	10	5	3	13	8	6	4	2	9	6
G	0	5	6	1	8	18	13	11	9	7	7
A	0	5	1	2	6	13	14	9	7	14	8
T	5	1	1	6	4	11	9	10	5	9	9
T	5	1	0	6	2	9	7	5	6	7	10
	1	2	3	4	5	6	7	8	9	10	

FIGURE 2.9. Local alignment matrix ($\hat{S}(i,j)$).

Chapter 3
Statistics of Local Sequence Alignment

"The patient is more alive then dead"

N. Nosov from "The Adventures of Neznaika and His Friends" [48]

3.1. Hypothesis Testing

Suppose we have to make a *binary* decision based on some evidence, where the term *binary* indicates two mutually excluding options. A theory that can help us with this task is called *hypothesis testing*. A formal model for statistical hypothesis testing was proposed by J. Neyman and E. S. Pearson in the late 1920s and 1930s. In hypothesis testing, the states of nature are partitioned into two sets or hypotheses. The goal of hypothesis testing is to decide which hypothesis is correct, i.e., which hypothesis contains the true state of nature. We need to distinguish between two hypothesis, so we call them the *null* hypothesis H_0 and the *alternative* hypothesis H_1. Traditionally, the null hypothesis is the one that we accept by default even if the evidences for both H_0 and H_1 are equivocal and H_1 is the hypothesis that requires compelling evidence in order to be accepted. Generally, H_0 is a mathematical model of randomness with respect to a particular set of observations. It gives a chance its privileged status in statistical theory. The purpose of most statistical tests is to determine whether the observed data can be explained by the null hypothesis. For this purpose P-values are used. In the case of database retrieval, E-values are used instead of P-values. The P-value is the probability of observing an effect as strong or stronger than you observed, given the null hypothesis, thus it answers the question "How likely is this effect to occur by chance?". The lower the P-value, the less likely the result, assuming the null hypothesis, the more significant the result, in the sense of statistical significance. In the case of database retrieval, i.e. when we list/retrieve the highest scoring sequences that are supposedly homologous with our query sequence, a P-value by itself is not informative enough. For example, let our null hypothesis be that our alignment comes from the random model R and the alternative hypothesis that our alignment is real, coming from the match model M. Consider a score that corresponds to a P-value 10^{-4}. On the face of it, this seems a very unlikely event. Roughly speaking, we would expect to see an alignment at least this good only once in every $10,000$ alignments. Now, if our database has a large size (say, $100,000$), we would still expect to see quite a few chance alignments that are as good as the one we consider. Thus, the E-value — the expected number of times that we would see a result at least as good as the one we have in a random database of the given size — is a better way to judge whether we have compelling evidence to accept our alternative hypothesis. To assess whether a given alignment constitutes evidence for homology, it helps to know how strong an alignment can be expected from chance alone. In this context, "chance" can mean the comparison of sequences that are generated randomly based upon a DNA or protein sequence model. In sequence similarity search, homology relationship is inferred based on P-values or its equivalent E-values. These values help to decide on two hypotheses: H_0 - an alignment score was obtained by chance, meaning that sequences are not homologous, and H_1 - the score is significant and sequences are homologous.

If a local alignment has score x, the P-value gives the probability that a local alignment having score x or greater is found by chance. The smaller the P-value, the greater the belief that the aligned sequences are homologous. Accordingly, two sequences are reported to be homologous if they are aligned *extremely well*, which serves as a compelling evidence.

The BLAST program (Basic Local Alignment Search Tool [4]) is the most widely used tool for homology search in DNA and protein database. It finds regions of local similarity between a query sequence and each database sequence. It also calculates the statistical significance of matches. It has been used by numerous biologists to reveal functional and evolutionary relationships between sequences and identify members of gene families. A P-value of 10^{-5} is often used as a cutoff for BLAST database search. It means that with a collection of random query sequences, only once in a hundred thousand of instances would an alignment with that score or greater occur by chance.

3.2. Random Variables and Random Processes

In order to understand the rigorous statistical theory implemented in BLAST we first introduce more facts from probability theory. Proofs and further details on this topic can be found in [20], [23], [43].

Consider a probability space $\{\Omega, \mathcal{F}, \mathbb{P}\}$ defined in Chapter 2. Sometimes we are more interested in the consequence of the random outcome of an experiment then in the experiment itself. When we assign real values to such consequences we can think of them as functions. Such functions are called *random variables.*

Definition 17. A *random variable* is a function $X : \Omega \to \mathbb{R}$ with the property that $\{\omega \in \Omega : X(\omega) \leq x\} \in \mathcal{F}$ for each $x \in \mathbb{R}$. Such a function is said to be *$\mathcal{F}$-measurable.*

The value taken by a random variable is subject to chance. We shall denote random variables by upper-case letters X, Y, and Z etc., and numerical values of a random variable by lowercase letters x, y, and z, etc. For a random variable X and a real number x, the expression $\{X \leq x\}$ denotes the event that X has a value less then or equal to x. The probability that the event occurs is denoted by $\mathbb{P}[X \leq x]$.

Definition 18. The *distribution function* of a random variable X is the function $F_X : \mathbb{R} \to [0, 1]$ given by $F_X(x) = \mathbb{P}[X \leq x]$.

There are two important classes of random variables: *discrete* and *continuous.*

Definition 19. The random variable X is called *discrete* if it takes values in some countable subset $\{x_1, x_2, \ldots\}$, of $\mathbb{R}$. The discrete random variable X has *probability mass function* $f : \mathbb{R} \to [0, 1]$ given by $f(x) = \mathbb{P}[X = x]$. The distribution and mass functions are related by

$$F_X(x) = \sum_{i : x_i \leq x} f(x_i).$$

Definition 20. An *indicator* $I_A(w)$ of an event A is the random variable, such that

$$I_A = \begin{cases} 1, \text{ if } w \in A \text{ occurs }, \\ 0, \text{ if } w \notin A. \end{cases}$$

We will now define some useful discrete distributions .

(1) **Bernoulli trials.** A random variable X takes values 1 with probability p and 0 with probability $q = 1 - p$. Values 1 or 0 can also be interpreted as the "success" or the "failure" of a trial. This random variable X is called *Bernoulli* random variable with a parameter p. Its probability mass function is given by

$$f(x) = \begin{cases} 1 - p, \text{ if } x = 0, \\ p, \text{ if } x = 1. \end{cases} \tag{3.2.1}$$

Bernoulli variables are often used, as indicator variables of events are clearly Bernoulli variables.

(2) **Binomial distribution.** We perform n independent Bernoulli trials X_1, $X_2, \ldots, X_n$ with probability of success p and count the total number of successes

$$Y = X_1 + X_2 + \ldots + X_n.$$

This random variable Y is called a *Binomial* random variable with parameters n and p. Its probability mass function is given by

$$f(k) = \binom{n}{k} p^k (1-p)^{n-k}, \ k = 0, 1, \ldots, n. \tag{3.2.2}$$

Note, that if $n = 1$ then Y is simply a Bernoulli trial.

(3) **Poisson distribution.** A *Poisson* random variable with parameter $\lambda > 0$ is a random variable Z with the Poisson probability mass function given by the formula

$$f(k) = \frac{\lambda^k}{k!} e^{-\lambda}, \ k = 0, 1, 2, \ldots. \tag{3.2.3}$$

Poisson variables and binomial variables are intimately connected. Let Y be a Binomial random variable with parameters n and p, and suppose that n is very large and p is very small. Recall, that $\mathbb{E}(Y) = np$. We will let $n \to \infty$ and $p \to 0$ in such a way that $np \to \lambda$, where λ is a positive constant. Then, the probability of $\{Y = k\}$ for $k = 0, 1, 2, \ldots.$ can be approximated by

$$\mathbb{P}[Y = k] = \binom{n}{k} p^k (1-p)^{n-k} \approx \frac{1}{k!}\left(\frac{np}{1-p}\right)^k (1-p)^n \to \frac{\lambda^k}{k!} e^{-\lambda}.$$

(4) **Geometric and Geometric-like distributions.** A *geometric* random variable is a random variable W with the geometric mass function given by the formula

$$f(k) = p(1-p)^{k-1},\ k = 1, 2, \ldots \tag{3.2.4}$$

$$F_W(x) = 1 - p^{x+1},\ x = 0, 1, \ldots. \tag{3.2.5}$$

Suppose that V is a random variable taking possible values $0, 1, 2, \ldots$ and $0 < p < 1$. If

$$\lim_{v \to \infty} \frac{1 - F_V(v)}{p^{v+1}} = C \tag{3.2.6}$$

for some fixed C, then the random variable V is said to be *geometric-like*.

Suppose we perform independent Bernoulli trials with parameter p at times $1, 2, \ldots.$ Let W be a *waiting time,* which is the time that elapses before the first success. Then $\mathbb{P}[W > k] = (1-p)^k$ and thus

$$\mathbb{P}(W = k) = \mathbb{P}(W > k-1) - \mathbb{P}(W > k) = p(1-p)^{k-1}.$$

W is a *geometric* random variable with the geometric mass function

$$f(k) = p(1-p)^{k-1},\ k = 1, 2, \ldots \tag{3.2.7}$$

The distribution function of the geometric random variable W with parameter p is

$$F_W(x) = 1 - p^{x+1},\ x = 0, 1, \ldots. \tag{3.2.8}$$

Suppose that V is a random variable taking possible values $0, 1, 2, \ldots$ and $0 < p < 1$. If

$$\lim_{v \to \infty} \frac{1 - F_V(v)}{p^{v+1}} = C \tag{3.2.9}$$

for some fixed C, then the random variable V is said to be *geometric-like*.

Definition 21. The random variable X is called *continuous* if its distribution function can be expressed as

$$F_X(x) = \int_{-\infty}^{x} f(u)du \; x \in \mathbb{R},$$

for some integrable function $f : \mathbb{R} \to [0, \infty]$ called the *probability density function* of X.

The most important continuous distribution in this chapter is called the *extreme-value* distribution. Since it plays the key role in statistics of sequence alignment we will define it and discuss its basic properties in the next section.

Definition 22. The *mean value*, or *expectation*, or *expected value* of the discrete random variable X with mass function f is defined to be

$$\mathbb{E}(X) = \sum_{x:f(x)>0} xf(x)$$

whenever this sum is absolutely convergent.

Definition 23. The *expectation* of a continuous random variable X with density function f is given by

$$\mathbb{E}(X) = \int_{-\infty}^{\infty} xf(x)dx$$

whenever this integral exists.

Theorem 3.1. *If X and $g(X)$ are continuous random variables then*

$$\mathbb{E}(g(X)) = \int_{-\infty}^{\infty} g(x)f_X(x)dx$$

We can think of the process of calculation expectations as a linear operator on the space of random variables.

Theorem 3.2. *The expectation operator* $\mathbb{E}$ *has the following properties:*

(a) *if* $X \geq 0$ *then* $\mathbb{E}(X) \geq 0$,

(b) *if* $X_1, \dots, X_n$ *are random variables with the same range* I*, then for any constants* $a_1, \dots, a_n \in \mathbb{R}$ *the following holds*

$$\mathbb{E}(\sum_{i=1}^{n} a_i X_i) = \sum_{i=1}^{n} a_i \mathbb{E}(X_i). \tag{3.2.10}$$

This important property is called the linearity *of the expectation.*

(c) *the random variable* 1*, taking the value* 1*, has expectation* $\mathbb{E}(1) = 1$.

Definition 24. Discrete random variables X and Y are *independent* if the events $\{X = x\}$ and $\{Y = y\}$ are independent for all x and y.

Definition 25. Continuous random variables X and Y are *independent* if the events $\{X \leq x\}$ and $\{Y \leq y\}$ are independent for all $x, y \in \mathbb{R}$.

Theorem 3.3. *If random variables* X *and* Y *are independent then*

$$\mathbb{E}(XY) = \mathbb{E}(X)\mathbb{E}(Y).$$

Definition 26. For any positive integer, $\mathbb{E}(X^m)$ is called the rth *moment* of a random variable X provided that the expectation (i.e. the corresponding sum or integral) exist.

Definition 27. The variance of a random variable X is defined as $Var(X) = E((X - E(X))^2)$. Consequently, the *variance* of a discrete random variable X is defined as

$$\text{Var}(X) = \sum_{x \in I} (x - \mathbb{E}(X))^2 \mathbb{P}[X = x], \tag{3.2.11}$$

where I is the range of X. Similarly, the *variance* of a continuous random variable Y is defined by

$$\mathrm{Var}(Y) = \int_{-\infty}^{\infty} (y - \mathbb{E}(Y))^2 f_Y(y)dy. \tag{3.2.12}$$

Definition 28. The *moment-generating function* (MGF) of a random variable X is written $M_X(t)$ and defined as

$$M_X(t) = \mathbb{E}(e^{tX}).$$

For a discrete random variable X having range I,

$$M_X(t) = \sum_{x \in I} e^{tx} \mathbb{P}[X = x].$$

For a continuous random variable Y having density function f_Y,

$$M_Y(t) = \int_{-\infty}^{\infty} e^{ty} f_Y(y)dy.$$

Any moments of a random variable can be expressed as an appropriate differentiation of its MGF. In particular, for any random variable X,

$$\mathbb{E}(X) = \left(\frac{dM_X(t)}{dt} \right)_{t=0}$$

and

$$Var(X) = \left(\frac{d^2 M_X(t)}{dt^2} \right)_{t=0} - \mathbb{E}^2(X) = \left(\frac{d^2 \ln M_X(t)}{dt^2} \right)_{t=0}.$$

The following property of MGFs plays a critical role in the scoring matrix and local alignment statistics.

Theorem 3.4. *Let X be a discrete random variable with $M_X(t)$ converging for all t in $(-\infty, \infty)$. Suppose that X takes at least one negative value and one positive value with nonzero probability and that $\mathbb{E}(X) \neq 0$. Then there exists a unique nonzero value θ such that $M_X(\theta) = 1$.*

Definition 29. A *random process* is a family of random variables $\{X(t), t \in T\}$ indexed by the parameter t, where t varies over an index set T. The index set T is calles the *parameter set* of the random process. The values assumed by $X(t)$ are called *states,* and the set of all possible values forms the *state space* of the random process.

We will define two important random processes: a random walk and a renewal process. In both processes the parameter set and the state space are discrete sets.

Definition 30. Suppose that $X_1, X_2, \ldots$ are independent random variables, each taking values $x_i \in \{-n, -n+1, \ldots, m-1, m\}$, for $i \geq 1$ and $n, m \in \mathbb{N}$ with respective probabilities

$$p_{-n}, p_{-n+1}, \ldots, p_{-1}, p_0, p_1, \ldots, p_{m-1}, p_m,$$

and consider the sum $S_n = \sum_{i=1}^{n} X_i$, then the sequence $S = \{S_i : i \geq 0\}$ where $S_0 = 0$ is called a *generalised random walk in one dimension* starting at the origin. When $X_1, X_2, \ldots$ take values 1 with probability p, and -1 otherwise, then the sequence $S = \{S_i : i \geq 0\}$ where $S_0 = 0$ is called a *simple random walk in one dimension* starting at the origin.

Definition 31. A *renewal process* $N = \{N(t) : t \geq 0\}$ is a process such that

$$N(t) = \max\{n : T_n \leq t\}$$

where $T_0 = 0, T_n = X_1 + X_2 + \ldots + X_n$ for $n \geq 1$, and $\{X_i\}$ is a sequence of independent identically distributed non-negative random variables.

We can think of a renewal process $N(t)$ as representing the number of occurrences of some event in the time interval $[0, t]$; the event in question might be the arrival of a person or particle, or a failure of a light bulb. We shall speak of T_n as the *time of nth arrival* and X_n as the *nth interarrival time.*

Consider an infinite sequence of repeated trials with possible outcomes $X_i, i = 1, 2 \ldots$ Let B be an event defined by an attribute of the finite sequences of possible outcomes X_i. We say that B occurs at the ith trial if the outcomes $X_1, X_2, \ldots, X_i$ have the attribute. H is a *recurrent event* if, under the condition the H occurs at the ith trial, we have that the probability that $X_1, X_2, \ldots, X_{i+k}$ have the attribute is equal to the product of the probabilities that $X_1, X_2, \ldots, X_i$ have the attribute and $X_{i+1}, X_{i+2}, \ldots, X_{i+k}$ have the attribute. For instance, the event that a success followed by failure is a recurrent event in a sequence of Bernoulli trials.

For a recurrent event H, we are interested in the following two probabilistic distributions:

$$u_j = \mathbb{P}[H \text{ occurs at the } i\text{th trial }], i = 1, 2, \ldots,$$

$$f_i = \mathbb{P}[H \text{ occurs for the first time at the } i\text{th trial }], i = 1, 2, \ldots,$$

Then the mean waiting time between two successive occurrences of H is

$$\mathbb{E}(H) = \sum_i i f_i. \tag{3.2.13}$$

Theorem 3.5. *If $\mathbb{E}(H)$ is finite, then $u_n \to \frac{1}{\mathbb{E}(H)}$ as $n \to \infty$.*

Define the indicator variable I_j for each j as

$$I_j = \begin{cases} 1, & \text{if } H \text{ occurs at the } j\text{th trial,} \\ 0, & \text{if it does not.} \end{cases}$$

Then, the sum

$$X_k = I_1 + I_2 + \ldots + I_k$$

denotes the number of occurrences of H in the first k trials. By the linearity property of the expectation, we have

$$\mathbb{E}(X_k) = \mathbb{E}(I_1) + \mathbb{E}(I_2) + \ldots + \mathbb{E}(I_k) = u_1 + u_2 + \ldots + u_k.$$

If $\mathbb{E}(H)$ is finite, by Theorem 3.5, we have

$$\mathbb{E}(X_N) \approx \frac{N}{\mathbb{E}(H)}. \tag{3.2.14}$$

Definition 32. The *Renewal Equation* is a convolution equation relating bounded sequences $\{\nu_i\}_{i\geq 0}$ and $\{b_i\}_{i\geq 0}$ of real numbers:

$$\nu_n = b_n + \sum_{k=1}^{n} f_k \nu_{n-k}, \ n = 0, 1, \ldots. \tag{3.2.15}$$

where $\{f_k\}_{k\geq 1}$ is the interarrival time distribution for the renewal process.

The following theorem provides conditions under which the sequence ν_n defined through convolution equation converges as n goes to infinity.

Theorem 3.6. *If $\sum_i b_i = B < \infty$, $\sum_i f_i = 1$, and $\xi = \sum_i i f_i < \infty$, and the greatest common divisor of indicies i such that $f_i \neq 0$ is 1, then*

$$\nu_n \longrightarrow \frac{B}{\xi}$$

as $n \to \infty$.

3.3. Ungapped Local Alignment Scores Statistics

3.3.1. Ungapped local alignment scores. To study the distribution of optimal local ungapped alignment scores, we use the random model R, introduced in a previous chapter. Recall, that within this model we assume that the two aligned sequences are made up of residues i that are drawn independently with different probabilities p_i. These probabilities define the *background frequency distribution* of the aligned sequences. We denote the score for aligning residues i and j by s_{ij}. If at least one score s_{ij} is positive and the expected score for aligning two randomly chosen residues is negative, i.e.

$$\mathbb{E}(s_{ij}) = \sum_{i,j} p_i p_j s_{ij} < 0, \tag{3.3.1}$$

the optimal local ungapped alignment scores approach an extreme value distribution when sequences are sufficiently long. Moreover, simple formulas are available for the corresponding parameters λ and μ [31], [30], [16].

The scale parameter λ is the unique positive number satisfying the following equation

$$\sum_{i,j} p_i p_j e^{\lambda s_{ij}} = 1. \tag{3.3.2}$$

Thus, by (3.3.2), λ depends on the scoring matrix (s_{ij}) and the background frequencies (p_i). It converts pairwise match scores to a probabilistic distribution $p_i p_j e^{\lambda s_{ij}}$.

The location parameter μ is given by

$$\mu = \frac{\log(Kmn)}{\lambda}, \tag{3.3.3}$$

where m and n are the lengths of aligned sequences and $K < 1$. K is considered as a space correcting factor because optimal local alignments cannot locate in all mn possible sites. It is analytically given by a geometrically convergent series, depending only on the (p_i) and (s_{ij})

For example, consider a fixed ungapped alignment between two sequences shown in *Figure* 3.1.

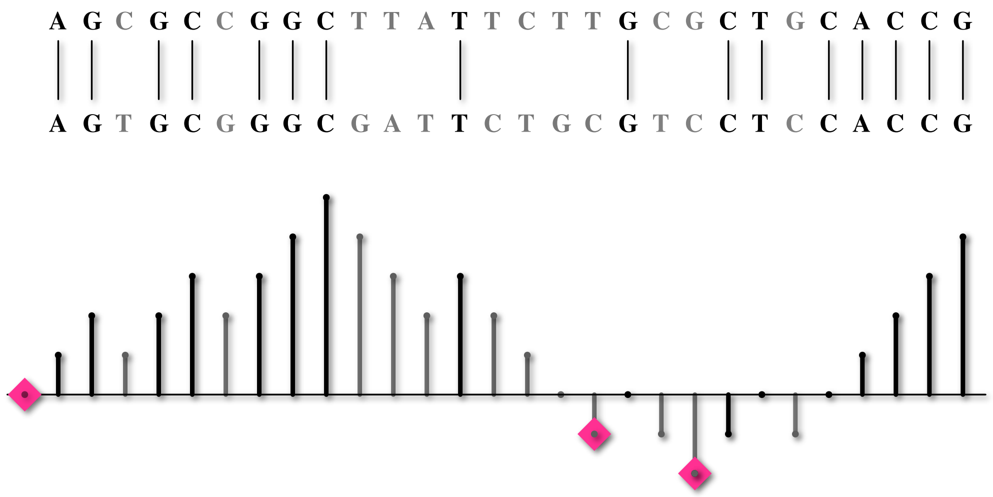

A G C G C C G G C T T A T T C T T G C G C T G C A C C G
A G T G C G G G C G A T T C T G C G T C C T C C A C C G

FIGURE 3.1. Ungapped alignment between two DNA sequences and its accumulative score [10].

We use X_j to denote the score of the aligned pair of residues at position j and consider the *accumulative score*

$$S_k = X_1 + X_2 + \ldots + X_k, k = 1, 2, \ldots.$$

Starting from the left, the accumulative score S_k is graphically represented below in *Figure* 3.1. A position j in the alignment is called a (*descending*) *ladder position* if the accumulative score S_j is lower then S_i for any $1 \leq i < j$. In *Figure* 3.1 the ladder positions are indicated by diamonds. Consider two consecutive ladder positions a and b, where $a < b$. For a position x between a and b, we define the *relative accumulative score* at x as

$$R_x = S_x - S_a,$$

which is the difference of the accumulative scores at the positions x and a.

In a random ungapped alignment, the relative accumulative score is a random variable that can be considered as a random walk process. For example, if a match score is s and it occurs with probability p, then R_x can be considered as the distance from 0 in a random walk that moves right with probability p or left with probability $1-p$ and stops at -1 on the state set $\{-1, 0, 1, \dots\}$. The state -1 is called the *absorbing* state, which is the state the walk once reaches and never leaves.

The local alignment between a ladder position where the highest relative accumulative score attains before the next ladder position gives a *maximal scoring segment* in the alignment. In 1992 Karlin and Dembo published the paper [31], where they derived the explicit limit distribution of maximal scoring segments. Their method was based on the estimation of the following two quantities:

(i) The probability distribution of the maximum value that the corresponding random walk ever achieves before stopping at the absorbing state, and

(ii) The mean number of steps before the corresponding random walk first reaches the absorbing state.

In sections 3.3.2, 3.3.3, and 3.3.4 we will provide basic results from the paper [31] with proofs.

3.3.2. Maximum Segment Scores. When two sequences are aligned, scores are taken from a substitution matrix, for example, BLOSUM62. Because such matrices are integer valued, the accumulative score performes a complicated random walk.

Consider a random walk that starts at 0 and whose possible step sizes are

$$-d, -d+1, \dots, -1, 0, 1, \dots, c-1, c$$

with respective probabilities

$$p_{-d}, p_{-d+1}, \dots, p_{-1}, p_0, p_1, \dots, p_{c-1}, p_c$$

such that

(i) $p_{-d} > 0, p_c > 0$, and $p_i \geq 0, -d < i < c$, and

(ii) the mean step size $\sum_{j=-d}^{c} jp_j < 0$.

We denote the score of the aligned pair at the ith position by X_i. Clearly, X_is are independent identically distributed (i.i.d.) random variables. Consider a random variable X with the same distribution as X_is. The moment generating function of X is

$$\mathbb{E}(e^{\theta X}) = \sum_{j=-d}^{c} p_j e^{j\theta}.$$

We consider the *accumulative scores*:

$$S_0 = 0,$$
$$S_j = \sum_{i=1}^{j} X_i, j = 1, 2, \ldots,$$

and partition the walk into non-negative *excursions* between the successive descending ladder positions in the path as shown in *Figure* 3.2:

$$K_0 = 0,$$
$$K_i = \min\{k | k \geq K_{i-1} + 1, S_k < S_{K_{i-1}}\}, i = 1, 2, \ldots. \tag{3.3.4}$$

Because the mean step size is negative, the $K_i - K_{i-1}$ are positive integer-valued i.i.d. random variables. Define Q_i to be the maximal score attained during the ith excursion between K_{i-1} and K_i, i.e.

$$Q_i = \max_{K_{i-1} \leq k < K_1} (S_k - S_{K_{i-1}}), i = 1, 2, \ldots. \tag{3.3.5}$$

The Q_is are non-negative i.i.d. random variables. We will estimate $\mathbb{P}[Q_1 \leq x]$ and $\mathbb{E}(K_1)$.

Define

$$t^+ = \min\{k | S_k > 0, S_i \leq 0, i = 1, 2, \ldots, k-1\}, \tag{3.3.6}$$

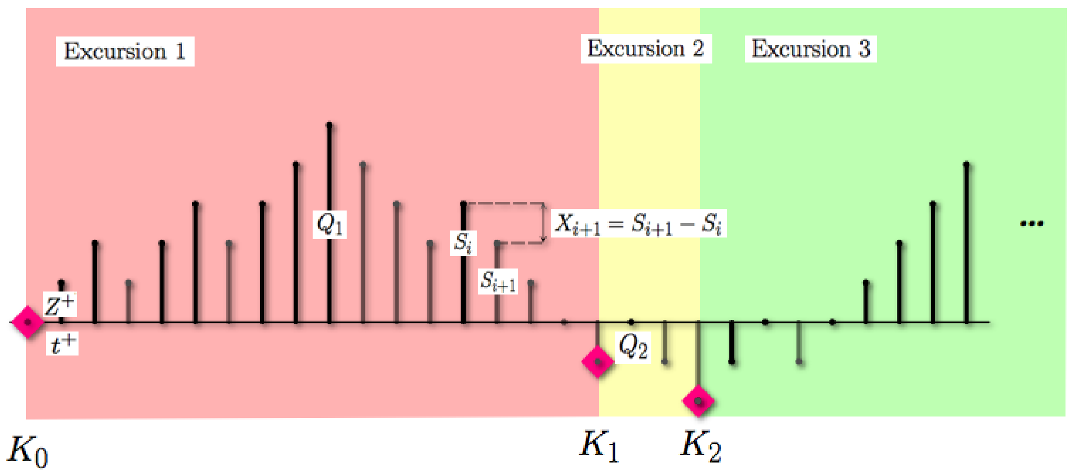


FIGURE 3.2. Random walk corresponding to the ungapped local alignment in *Figure* 3.1.

and set $t^+ = \infty$ if $S_i \leq 0$ for all i. Then t^+ is the *stopping time* of the first positive accumulative score. We define

$$Z^+ = S_{t^+}, t^+ < \infty \tag{3.3.7}$$

Lemma 3.7. *With notations defined above,*

$$\mathbb{E}(K_1) = \exp\left\{\sum_{k=1}^{\infty} \frac{1}{k}\mathbb{P}[S_k \geq 0]\right\}, \tag{3.3.8}$$

and

$$1 - \mathbb{E}(e^{\theta Z^+}; t^+ < \infty) = (1 - \mathbb{E}(e^{\theta X})) \exp\left\{\sum_{k=1}^{\infty} \frac{1}{k}\mathbb{E}(e^{\theta S_k}; S_k \leq 0)\right\}. \tag{3.3.9}$$

PROOF. The formula (3.3.8) is symmetric to formula (7.16) in Theorem 7.3 in [20] on page 416, which is proved for strict ascending ladder positions.

Because the mean step size is negative in our case, we have that

$$1 - \mathbb{E}(e^{\theta Z^+}; t^+ < \infty) = \exp\left\{-\sum_{k=1}^{\infty} \frac{1}{k}\mathbb{E}(e^{\theta S_k}; S_k > 0)\right\}. \tag{3.3.10}$$

By using the same argument as in the proof of a result due to Baxter (see Theorem 4.1 in [43] on pages 220-221). Because

$$\mathbb{E}(e^{\theta S_k}; S_k > 0) + \mathbb{E}(e^{\theta S_k}; S_k \le 0) = \mathbb{E}(e^{\theta S_k}) = \Big(\mathbb{E}(e^{\theta X})\Big)^k$$

for any k and

$$\log(1-y) = -\sum_{i=1}^{\infty} \frac{1}{i} y^i$$

for $0 \le y < 1$, the equation (3.3.10) becomes

$$\begin{aligned} 1 - \mathbb{E}(e^{\theta Z^+}; t^+ < \infty) &= \exp\Big\{-\sum_{k=1}^{\infty} \frac{1}{k}\Big(\mathbb{E}(e^{\theta X})\Big)^k + \sum_{k=1}^{\infty} \frac{1}{k}\mathbb{E}(e^{\theta S_k}; S_k \le 0)\Big\} \\ &= \exp\Big\{\ln\Big(1 - \mathbb{E}(e^{\theta X})\Big)\Big\} \exp\Big\{\sum_{k=1}^{\infty} \frac{1}{k}\mathbb{E}(e^{\theta S_k}; S_k \le 0)\Big\} \\ &= \Big(1 - \mathbb{E}(e^{\theta X})\Big) \exp\Big\{\sum_{k=1}^{\infty} \frac{1}{k}\mathbb{E}(e^{\theta S_k}; S_k \le 0)\Big\}. \end{aligned}$$

□

Setting $\theta = 0$, the equation (3.3.10) reduces to

$$1 - \mathbb{P}[t^+ < \infty] = \exp\Big\{-\sum_{k=1}^{\infty} \frac{1}{k}\mathbb{P}[S_k > 0]\Big\}. \tag{3.3.11}$$

By the Theorem 3.4 the equation $\mathbb{E}(e^{\theta X}) = 1$ has the unique positive root. We will denote this root by λ, i.e.

$$\sum_{j=-d}^{c} p_j e^{j\lambda} = 1, \tag{3.3.12}$$

then (3.3.9) implies

$$1 = \mathbb{E}\Big(e^{\lambda Z^+}; t^+ < \infty\Big) = \sum_{k=1}^{\infty} e^{\lambda k}\mathbb{P}[Z^+ = k]. \tag{3.3.13}$$

Because the mean step size is negative, the distribution function

$$S(y) = \mathbb{P}[\max_{k\ge 0} S_k \le y] \tag{3.3.14}$$

is finite. Moreover, it satisfies the renewal equation defined by (3.2.15)

$$\begin{aligned} S(y) &= S(-1) + (S(y) - S(-1)) \\ &= (1 - \mathbb{P}[t^+ < \infty]) + \sum_{k=0}^{y} \mathbb{P}[Z^+ = k] S(y-k) \end{aligned} \tag{3.3.15}$$

for any $0 \leq y < \infty$. Define

$$V(y) = (1 - S(y))e^{\lambda y} \tag{3.3.16}$$

V(y) satisfies the renewal equation

$$V(y) = (\mathbb{P}[t^+ < \infty] - \mathbb{P}[Z^+ \leq y])e^{\lambda y} + \sum_{k=0}^{y} e^{\lambda k} \mathbb{P}[Z^+ = k] V(y-k). \tag{3.3.17}$$

By equation (3.3.13), $e^{\lambda k}\mathbb{P}[Z^+ = k]$ is a probability distribution.

Using the Theorem 3.6 known as the *Key Renewal Theorem* we have that

$$\lim_{y \to \infty} V(y) = \frac{\sum_{y=0}^{\infty} e^{\lambda y}(\mathbb{P}[t^+ < \infty] - \mathbb{P}[Z^+ \leq y])}{\mathbb{E}(Z^+ e^{\lambda Z^+}; t^+ < \infty)}.$$

By (3.3.13), multiplying the numerator by $e^{\lambda} - 1$ yields $1 - \mathbb{P}[t^+ < \infty]$ because the step sizes that have nonzero probability have 1 as their greatest common divisor. Hence,

$$\lim_{y \to \infty} V(y) = \frac{1 - \mathbb{P}[t^+ < \infty]}{(e^{\lambda} - 1)\mathbb{E}(Z^+ e^{\lambda Z^+}; t^+ < \infty)}.$$

Recall that $Q_1 = \max_{0 \leq k < K_1} S_k$ and define

$$F(x) = \mathbb{P}[Q_1 \geq x], 0 \geq x < \infty. \tag{3.3.18}$$

For any positive integer h, define

$$\sigma(0, h) = \min\{k | S_k \notin [0, h]\}.$$

By definition of $S(y)$ in (3.3.14) and the law of probabilities, expanding according to the outcome of $\sigma(0,y)$ (either $S_{\sigma(0,h)} > y$ or $S_{\sigma(0,h)} < 0$) yields

$$1 - S(h) = 1 - F(h) + \sum_{k=1}^{\infty} (1 - S(h+k)) \mathbb{P}[Q_1 < h \text{ and } S_{\sigma(0,h)} = -k].$$

We define Z^- to be the first negative accumulative score, i.e.,

$$Z^- = S_{t^-}, t^- = \min\{k | S_k < 0, S_i \geq 0, i = 1, 2, \ldots, k-1\}.$$

Multiplying by $e^{\lambda h}$ throughout the last equation and taking limit $h \to \infty$, we obtain

$$\begin{aligned} \lim_{h\to\infty} e^{\lambda h}(1 - F(h)) &= \lim_{h\to\infty} V(h)\left(1 - \sum_{k=1}^{\infty} e^{-\lambda k} \mathbb{P}[S_{\sigma(0,\infty)} = -k]\right) \\ &= \frac{(1 - \mathbb{P}[t^+ < \infty])\left(1 - \mathbb{E}\left(e^{\lambda Z^-}\right)\right)}{(e^\lambda - 1)\mathbb{E}(Z^+ e^{\lambda Z^+}; t^+ < \infty)} \end{aligned} \tag{3.3.19}$$

from (3.3.16).

Combining (3.3.8) and (3.3.11) gives

$$1 - \mathbb{P}[t^+ < \infty] = \frac{\{\sum_{k=1}^{\infty} \frac{1}{k} \mathbb{P}[S_k = 0]\}}{\mathbb{E}(K_1)}.$$

Differentiating (3.3.9) with respect to θ and setting $\theta = \lambda$ afterwards shows

$$\begin{aligned} \mathbb{E}\left(Z^+ e^{\lambda Z^+}; t^+ < \infty\right) &= \mathbb{E}\left(X e^{\lambda X}\right) \exp\left\{\sum_{k=1}^{\infty} \frac{1}{k} \mathbb{E}\left(e^{\lambda S_k}; S_k \leq 0\right)\right\} \\ &= \mathbb{E}\left(X e^{\lambda X}\right) \exp\left\{\sum_{k=1}^{\infty} \frac{1}{k} \mathbb{E}\left(e^{\lambda S_k}; S_k < 0\right) + \mathbb{P}[S_k = 0]\right\} \\ &= \frac{\mathbb{E}\left(X e^{\lambda X}\right) \exp\left\{\sum_{k=1}^{\infty} \frac{1}{k} \mathbb{P}[S_k = 0]\right\}}{1 - \mathbb{E}(e^{\lambda Z^-})}. \end{aligned}$$

Setting

$$C = \frac{(1 - \mathbb{P}[t^+ < \infty])\left(1 - \mathbb{E}\left(e^{\lambda Z^-}\right)\right)}{\mathbb{E}(Z^+ e^{\lambda Z^+}; t < \infty)} = \frac{\left(1 - \mathbb{E}\left(e^{\lambda Z^-}\right)\right)^2}{\mathbb{E}(X e^{\lambda X})\mathbb{E}(K_1)} \tag{3.3.20}$$

we have proved the folowing theorem:

Theorem 3.8. *Assume the step sizes that have nonzero probability do not have non-trivial common divisors. With notations defined above,*

$$\lim_{h\to\infty}(1-F(h))e^{\lambda h} = \frac{C}{e^{\lambda}-1}, \tag{3.3.21}$$

where λ is defined in (3.3.12).

Consider a random ungapped alignment $\mathcal{A}$. Q_1 defined for the random walk corresponding to $\mathcal{A}$ is equal to a maximal seqment score of it. Hence, $1 - F(s)$ denotes the probability of a maximal segment having score s or more in $\mathcal{A}$. By Theorem 3.8, the maximal segment scores have a geometric-like distribution as defined in (3.2.9).

3.3.3. E-value and P-value estimation. Here, we continue the derivation of formulas for the estimation of the tail distribution of maximal segment scores in ungapped alignment.

Consider the successive excursions with associated local maxima Q_k defined in (3.3.5). Define

$$M(K_m) = \max_{k\le m} Q_k \tag{3.3.22}$$

and

$$C' = \frac{C}{e^{\lambda}-1}. \tag{3.3.23}$$

Lemma 3.9. *With notation defined above,*

$$\liminf_{m\to\infty} \mathbb{P}\left[M(K_m) \le \frac{\ln m}{\lambda} + y\right] \ge \exp\left\{-C'e^{\lambda-\lambda y}\right\},$$

$$\limsup_{m\to\infty} \mathbb{P}\left[M(K_m) \le \frac{\ln m}{\lambda} + y\right] \le \exp\left\{-C'e^{-\lambda y}\right\}.$$

Proof. Because $Q_1, Q_2, \ldots, Q_m$ are i.i.d. random variables,

$$\mathbb{P}[M(K_m) \le x] = (F(x))^m. \tag{3.3.24}$$

for any $0 < x \leq \infty$, where $F(x)$ is the distribution function of Q_1 and is defined in (3.3.18). For any real number $x < \infty$,

$$1 - F(\lceil x \rceil) \leq 1 - F(x) \leq 1 - F(\lfloor x \rfloor).$$

Hence,

$$\liminf_{x\to\infty}(1 - F(x))e^{\lambda x} \geq \liminf_{x\to\infty}(1 - F(\lceil x \rceil))e^{\lambda\lceil x \rceil} = \frac{C}{e^{\lambda} - 1} = C'$$

On the other hand, because $\limsup_{x\to\infty} x - \lfloor x \rfloor = 1$,

$$\limsup_{x\to\infty}(1 - F(x))e^{\lambda x} \leq \lim_{x\to\infty}(1 - F(\lfloor x \rfloor))e^{\lfloor x \rfloor}e^{\lambda x - \lfloor x \rfloor} \leq \frac{C}{e^{\lambda} - 1}e^{\lambda} = C'e^{\lambda}.$$

Replacing x by $\ln(m)/\lambda + y$, (3.3.24) becomes

$$\mathbb{P}[M(K_m) \leq \ln(m)/\lambda + y] = (F(\ln(m)/\lambda + y))^m = \frac{1}{\exp\{-m\ln(F(\ln(m)/\lambda + y))\}}.$$

Using

$$\lim_{z\to 1}\frac{\ln z}{z - 1} = 1,$$

we obtain

$$\begin{aligned}\limsup_{m\to\infty}\mathbb{P}[M(K_m) \leq \ln(m)/\lambda + y] &= \limsup_{m\to\infty}\frac{1}{\exp\{-m\ln(F(\ln(m)/\lambda + y))\}}\\ &= \frac{1}{\liminf_{m\to\infty}\exp\{m(1 - F(\ln(m)/\lambda + y))\}}\\ &\leq \frac{1}{\liminf_{m\to\infty}\exp\{mC'e^{-(\ln(m)+\lambda y)}\}}\\ &= \exp\left\{-C'e^{-\lambda y}\right\}.\end{aligned}$$

Similarly,

$$\begin{aligned}\liminf_{m\to\infty} \mathbb{P}[M(K_m) \le \ln(m)/\lambda + y] &= \liminf_{m\to\infty} \frac{1}{\exp\{-m\ln(F(\ln(m)/\lambda + y))\}} \\ &= \frac{1}{\limsup_{m\to\infty} \exp\{m(1 - F(\ln(m)/\lambda + y))\}} \\ &\ge \frac{1}{\limsup_{m\to\infty} \exp\{mC'e^{\lambda}e^{-(\ln(m)+\lambda y)}\}} \\ &= \exp\Big\{-C'e^{\lambda}e^{-\lambda y}\Big\}.\end{aligned}$$

Hence, the lemma is proved. □

Now consider the accumulative score walk W corresponding to an ungapped alignment of length n. Let

$$A = \mathbb{E}(K_1), \tag{3.3.25}$$

which is the mean distance between two successive ladder positions in the walk. Then the mean number of ladder positions is approximately $\frac{n}{A}$ when n is large. Ignoring edge effects, we derive the following asymptotic bounds from Lemma 3.9 by setting $y = y' + \frac{\ln A}{\lambda}$ and $m = \frac{n}{A}$:

$$\exp\Big\{-\frac{C'e^{\lambda}}{A}e{-}\lambda y'\Big\} \le \mathbb{P}\Big[M(n) \le \frac{\ln n}{\lambda} + y'\Big] \le \exp\Big\{-\frac{C'}{A}e^{-\lambda y'}\Big\}. \tag{3.3.26}$$

Set

$$K = \frac{C'}{A}. \tag{3.3.27}$$

Replacing y' by $(\ln(K) + s)/\lambda$, inequality (3.3.26) becomes

$$\exp\Big\{-e^{\lambda - s}\Big\} \le \mathbb{P}[M(n) \le \ln(Kn) + s/\lambda] \le \exp\{-e^{-s}\},$$

or equivalently

$$\exp\Big\{-e^{\lambda - s}\Big\} \le \mathbb{P}[\lambda M(n) - \ln(Kn) \le s] \le \exp\{-e^{-s}\}. \tag{3.3.28}$$

In the BLAST theory, the expression

$$Y(n) = \lambda M(n) - \ln(Kn)$$

is called the *normalized score* of the alignment. Hence, the P-value corresponding to an observed value s of the normalized score is

$$P\text{-value} \approx 1 - \exp\{-e^{-s}\}. \tag{3.3.29}$$

3.3.4. The Number of High-Scoring Segments. By Theorem 3.8 and equation (3.3.23), the probability that any maximal-scoring segment has score s or more is approximately $C'e^{-\lambda s}$. By (3.3.25), to a close approximation there are $\frac{N}{A}$ maximal scoring segments in a fixed alignment of N columns as discussed in the previous section. Hence, the expected number of the maximal-scoring segments with score s or more is approximately

$$\frac{NC'}{A}e^{-\lambda s} = NKe^{-\lambda s}, \tag{3.3.30}$$

where K is defined in (3.3.27).

3.3.5. Karlin-Altschul Sum Statistic. When two sequences are aligned, insertions and deletions can break a long alignment into several parts. In this case, it is useful to consider the scores of the multiple highest-scoring segments.

Denote the r disjoint highest segment scores as

$$M^{(1)}, M^{(2)}, \ldots, M^{(r)}$$

in an alignment with n columns. We consider the normalized scores

$$S^{(i)} = \lambda M^{(i)} - \ln(Kn), i = 1, 2, \ldots, r. \tag{3.3.31}$$

Karlin and Altschul in 1993 [30] showed that the limiting joint density $f_S(x_1, x_2, \ldots, x_r)$ of $S = (S^{(1)}, S^{(2)}, \ldots, S^{(r)})$ is

$$_S(x_1, x_2, \ldots, x_r) = \exp\left(-e^{-x_r} - \sum_{k=1}^{r} x_k\right) \tag{3.3.32}$$

in the domain $x_1 \geq x_2 \geq \ldots \geq x_r$.

Assessing multiple highest-scoring segments is more involved that it might first appear. Suppose, for example, comparison X reports two highest scores 108 and 88, whereas comparison Y reports 99 and 90. One can say that Y is not better that X, because its high score is lower than of X. But neither is X considered better, because the second high score is lower than that of Y. The natural way to rank all possible results is to consider the sum of the normalized scores of the r highest scoring segments

$$S_{n,r} = S^{(1)} + S^{(2)} + \ldots + S^{(r)} \tag{3.3.33}$$

as suggested by Karlin and Altschul. This sum is now called the $Karlin - Altschul$ *sum statistic.*

Theorem 3.10. *The limiting density function of Karlin-Altschul sum $S_{n,r}$ is*

$$f_{n,r}(x) = \frac{e^{-x}}{r!(r-2)!} \int_0^\infty y^{r-2} \exp(-e^{(y-x)/r}) dy. \tag{3.3.34}$$

Integration $f_{n,r}(x)$ from t to ∞ gives the tail probability that $S_{n,r} \geq r$. This resulting double integral can be easily calculated numerically. Asymptotically, the tail probability is well approximated by the formula

$$\mathbb{P}[S_{n,r} \geq r] = \frac{e^{-t} t^{r-1}}{r!(r-1)!}. \tag{3.3.35}$$

especially when $t > r(r-1)$. This is the formula used in the BLAST program for calculation P-value when multiple highest-scoring segments are reported. We refer the reader to [30] for proofs.

3.3.6. Local ungapped alignment general case. The statistic of previous sections considered the maximal scoring segments in a fixed ungapped alignment. In practice, we often need to perform a database search, where we look for all good matches between a query sequence and the sequences in a database. Thus, it is important to consider a general problem of calculation the statistical significance of a local ungapped alignment between two sequences. The sequences in a highest-scoring local ungapped alignment between two sequences is called the *maximal-scoring segment pairs* (MSP).

Consider two sequences of length n_1 and n_2. To find the MSPs of the sequences, we have to consider all $n_1 + n_2 - 1$ possible ungapped alignments between the sequences. Each such alignment yields a random walk as that studied the Introduction section to this chapter. Because the n_1+n_2-1 corresponding random walks are not independent, it is much more involved to estimate the mean value of the maximum segment score. The theory for this general case was developed by Dembo et al. We will provide their major results and refer the interested reader to the papers [15], [16] for proofs.

Consider the sequences $x_1 x_2 \ldots x_{n_1}$ and $y_1 y_2 \ldots y_{n_2}$ where x_i and y_j are residues. we use $s(x_i, y_j)$ to denote the score for aligning residues x_i and y_j. The optimal local ungapped alignment score S_{max} from the comparison of the sequences is

$$S_{max} = \max_{\Delta \leq \min\{n_1, n_2\}} \max_{\substack{i \leq n_1 - \Delta \\ j \leq n_2 - \Delta}} \sum_{\ell=1}^{\Delta} s(x_{i+\ell}, x_{j+\ell}).$$

Suppose the sequences are random and independent: x_i and y_j follows the same distribution. The random variable S_{max} has the following tail probability distribution:

$$\mathbb{P}[S_{max} > \frac{1}{\lambda} \log(n_1 n_2) + y] \approx 1 - e^{-Ke^{-\lambda y}}, \tag{3.3.36}$$

where λ is given by equation (3.3.2) and K is given by equation (3.3.27) with n replaced by $n_1 n_2$. The mean value of S_{max} is approximately

$$\frac{1}{\lambda} \log(K n_1 n_2). \tag{3.3.37}$$

If the definition of the normalized score S is modified as

$$S = \lambda S_{max} - \log(K n_1 n_2), \tag{3.3.38}$$

the P-value corresponding to an observed value s of S is

$$P\text{-value} \approx 1 - e^{-(e^{-s})}. \tag{3.3.39}$$

The expected number of the MSPs scoring s or more is

$$K n_1 n_2 e^{-\lambda s} \tag{3.3.40}$$

The above theory is developed under several conitions and only applies, for example, when the sequences are sufficiently long and the aligned sequences grow at similar rate. But empirical studies show that the above theory carries over essentially unchanged after edge effect is made in practical cases in which the aligned sequences are only a few hundred of base pairs long.

3.3.7. Edge Effects. When the aligned sequences have finite lengths n_1 and n_2, optimal local ungapped alignment is less likely to appear at the end of a sequence. As a result, the optimal local alignment score will be less than that predicted by theory. This is called the *edge effect.* Edge effects have to be taken into account by subtracting from n_1 and n_2 the mean length of MSPs.

Let $\mathbb{E}(L)$ denote the mean length of a MSP. Then effective lengths for the sequence compared are

$$n_i' = n_i - \mathbb{E}(L), i = 1, 2. \tag{3.3.41}$$

The normalized score (3.3.38) becomes

$$S' = \lambda S_{max} - \log(K n_1' n_2'). \tag{3.3.42}$$

The expected number of MSPs scoring s or more given in (3.3.40) is adjusted as

$$Kn_1'n_2'e^{-\lambda s}. \tag{3.3.43}$$

Given that the score of a MSP is denoted by s, the mean length $\mathbb{E}(L)$ of this MSP is obtained from dividing s by the expected score of aligning a pair of residues:

$$\mathbb{E}(L) = \frac{s}{\sum_{ij} q_{ij}s_{ij}}, \tag{3.3.44}$$

where q_{ij} is the target frequency at which we expet to see residue i aligned with residue j in the MSP, and s_{ij} is the score for aligning i and j. With the value of λ, the s_{ij} and the background frequencies q_i and q_j in hand, q_{ij} can be calculated as

$$q_{ij} \approx p_i p_j e^{\lambda s_{ij}}. \tag{3.3.45}$$

Simulation performed by Altschul and Gish [3] show that the values calculated from (3.3.44) are often larger than the empirical mean lengths of MSPs especially when n_1 and n_2 are in the range from 10^2 to 10^3 [3]. Accordingly, the effective lengths defined by (3.3.41) might lead to P-value estimates less than the correct values. The current version of BLAST calculates empirically the mean length of MSPs in database search [3].

3.4. Gapped Local Alignment Scores Statistics

3.4.1. Distribution of Gapped Local Alignment Scores. The explicit distribution of local similarity scores in the case of a gapped local alignment remains unknown. In many empirical studies [2], [3], [33], [38], [42], [44] it was shown that under certain conditions on the scoring matrix and gap penalty costs gapped local alignment scores asymptotically follow the Gumble distribution

$$\mathbb{P}[S \geq x] = 1 - \exp(-e^{-\lambda(x-\mu)}), \tag{3.4.1}$$

where parameters λ and μ can be effectively estimated through simulations.

Most empirical studies focus on the statistical distribution of the scores of optimal local alignments with affine gap costs. Consider a pairwise alignment score S_n, where n indicates the length of the alignment. It was observed that the smaller gap penalty which guarantees the expected score of each aligned pair to be positive causes S_n to grow linearly in n. Whereas, the larger gap penalty causes the expected score to be negative, so that the probability of local alignment having a positive score decays exponentially fast in its length, i.e. S_n grows like $\log n$. In this case local alignments of positive score become rare events, and as shown in [29], [5], such local alignment scores approach an extreme value type-I distribution. The effect of having the linear growth of S_n changing into the logarithmic as a result of increasing the gap cost is called the *phase transition*. In order to make the extreme value theory applicable we want our scoring scheme to be in the logarithmic phase. For example, empirical studies show that if BLOSUM62 matrix is used, the affine gap cost parameters (o, e) must satisfy $o \geq 9$ and $e \geq 1$ [3].

3.4.2. Empirical estimation of λ and μ. It follows from (3.4.1), that

$$\mathbb{P}[S < x] = \exp(-e^{-\lambda(x-\mu)}),$$

and

$$\ln[-\ln \mathbb{P}(S < x)] = -\lambda x + \lambda\mu. \tag{3.4.2}$$

The first method for estimating λ and μ, the *direct method*, was introduced by Altschul and Gish [3].

In order to estimate parameters λ and μ, we generate N pairs $\{X_i[0, L], Y_i[0, L]\}_{i=1}^N$ of random protein sequences of length L using *Robinson* & *Robinson* [40] amino acid frequencies, see *Table* 3.1, perform local alignment for each pair using BLOSUM62 and the affine gap penalty $\Delta(k) = o + ek$, $o \geq 9$, $k \geq 1$, and collect the optimal local alignment scores $\{\hat{M}_i(L, L)\}_{i=1}^N$. We organize the resulting data into the histogram $\{k, f_k, p_k, \ell_k\}$ for $k = 0, 1, 2, \ldots, n$ where

$$f_k = \mathbb{P}(\hat{M} = k) = \frac{\text{the number of pairs } \{X_i, Y_i\}, \text{ such that } \hat{M}_i = k}{N},$$

$$p_k = \mathbb{P}(\hat{M} < k) = \sum_{i=0}^{k-1} \mathbb{P}(\hat{M} = i),$$

and

$$\ell_k = \ln(-\ln p_k).$$

Then it follows from (3.4.2) that λ and μ can be estimated by the linear regression

$$\lambda = -\frac{(n+1)\sum_{i=0}^n i\ell_i - \sum_{i=0}^n i \sum_{i=0}^n \ell_i}{(n+1)\sum_{i=0}^n i^2 - (\sum_{i=0}^n i)^2}, \tag{3.4.3}$$

$$\mu = \frac{1}{n+1}\left(\frac{1}{\lambda}\sum_{i=0}^n \ell_i + \sum_{i=0}^n i\right). \tag{3.4.4}$$

3.4.3. Computer Simulations and Results. To answer the question of how well parameters λ and μ can be estimated by the method described in the previous section, we performed 4 experiments for different values of parameter $L = \{40, 100, 200, 400\}$. In each experiment the sequences were generated using *Robinson* & *Robinson* frequencies, the number of generated pairs was $N = 10^6$, and BLOSUM62

Amino Acid	**R&R counts**
Alanine	35155
Arginine	23105
Asparagine	20212
Aspartic acid	24161
Cycteine	8669
Glutamic acid	19208
Glutamine	28354
Glycine	33229
Histidine	9906
Isoleucine	23161
Leucine	40625
Lysine	25872
Methionine	10101
Phenylalanine	17367
Proline	23435
Serine	32070
Threonine	26311
Tryptophan	5990
Tyrosine	14488
Valine	29012

TABLE 3.1. Robinson & Robinson counts among total 450431 amino acids [40].

with the affine gap $\Delta(k) = -11 - k$, were used for scoring. In all four cases we were able to fit the simulations data to the extreme value type-I distribution (3.4.1) with estimated parameters λ and μ. The estimated values for λ and μ are summarized in *Table* 3.2. Our results are comparable with the results obtained with more advanced methods by Hartmann [25], see *Table* 3.3. The details for $L = 40$, $L = 100$, $L = 200$, and $L = 400$ are shown in *Figure* 3.4, *Figure* 3.6, *Figure* 3.8, *Figure* 3.10, *Figure* 3.5, *Figure* 3.7, *Figure* 3.9, *Figure* 3.11.

L	λ	μ
40	0.348465	15.619
100	0.307664	21.8939
200	0.293792	26.9246
400	0.282713	31.9884

TABLE 3.2. Simulations results.

L	λ	μ
40	0.355(5)	15.35
100	0.304(2)	21.67(4)
400	0.280(3)	32.01(3)

TABLE 3.3. Known estimates for λ and μ by Hartmann [25].

In probability density plots, see *Figure* 3.4, *Figure* 3.6, *Figure* 3.8, *Figure* 3.10, the error was calculated as a difference between simulated relative frequencies and corresponding Gumbel probability density estimates, i.e.

$$\text{Error} = f_{\text{SIM}} - f_{\text{EST}}.$$

We observed, that the range of the error decreases with the increase of the sequence length, suggesting that a better fit can be obtained in simulations with longer sequences. Also, in all four cases, the shape of the error graph reflects a small shift of estimated probabilities to the left, suggesting that the parameter μ must be slightly larger then its estimates.

In probability distribution plots, see *Figure* 3.5, *Figure* 3.7, *Figure* 3.9, and *Figure* 3.11, the relative error was calculated

$$\text{Error} = \frac{P_{\text{SIM}} - P_{\text{EST}}}{P_{\text{SIM}}}.$$

In all four cases the relative error graphs show fluctuations in the tail. This is due to the fact that as the largest alignment score within each simulation together with a few scores close to the largest score are extremely rare events. When scores approach the largest score, the number of counts is not strictly decreasing, see *Figure* 3.3, as opposed to the density function of extreme value distribution, which is strictly decreasing on its tail.

This tail behavior is a major limitation of estimation by regression, since we lose precision in the most useful area of the distribution, i.e. where P-values are small ($< 10^{-5}$). The only way to improve estimation by this method is to increase the length of sequences L and the number of trials N, which will require tremendous computational efforts. Moreover, sequence databases tend to be so large that often scientists are interested in very small P-values ($\leq 10^{-40}$.) In this case the regression method is not applicable at all. More advanced methods were developed to simulate probability distribution of alignment scores in the regions where the events are very unlikely, see [25] for example.

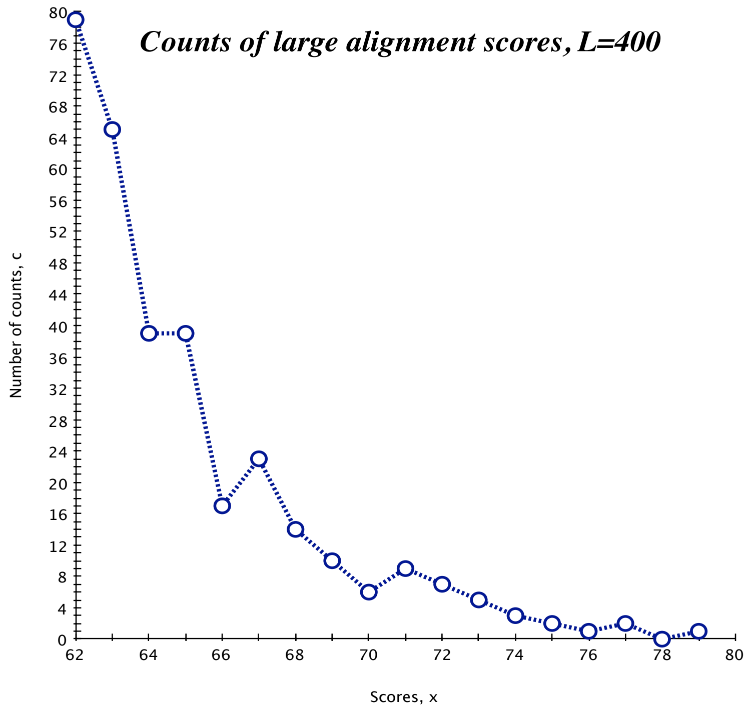


FIGURE 3.3. When alignment scores approach the largest score, the number of counts is not strictly decreasing.

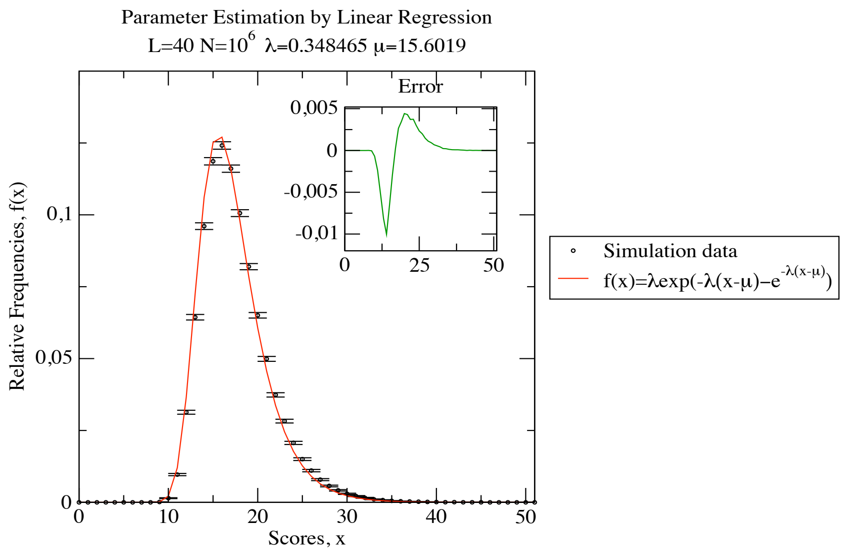


FIGURE 3.4. Probability density plot of simulated and estimated data for sequences of length $L = 40$.

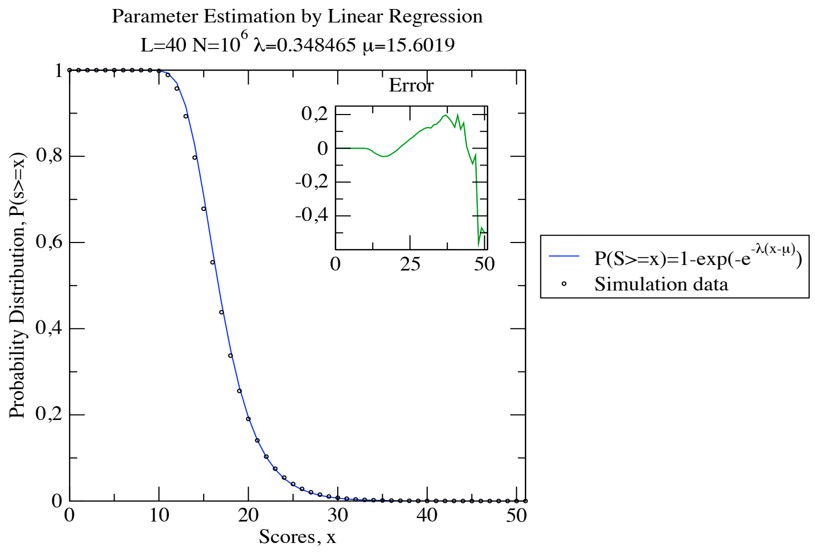


FIGURE 3.5. Probability distribution plot of simulated and estimated data for sequences of length $L = 40$.

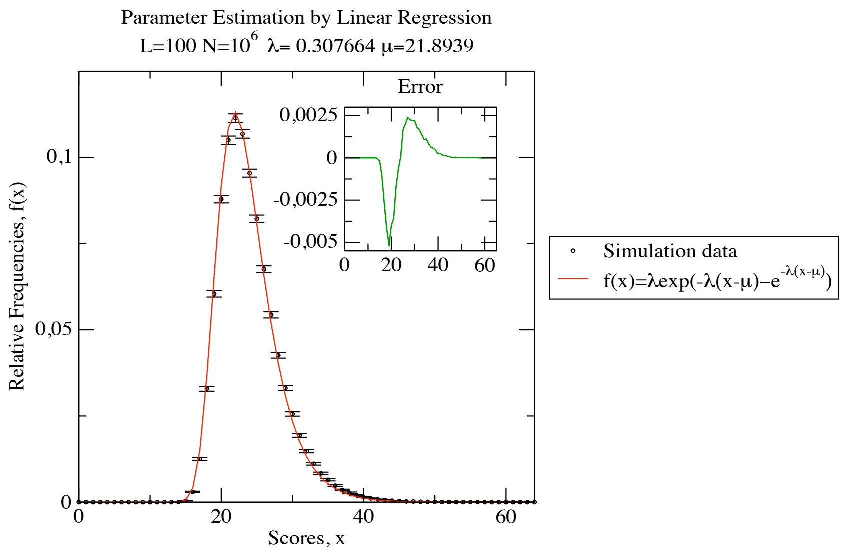


FIGURE 3.6. Probability density plot of simulated and estimated data for sequences of length $L = 100$.

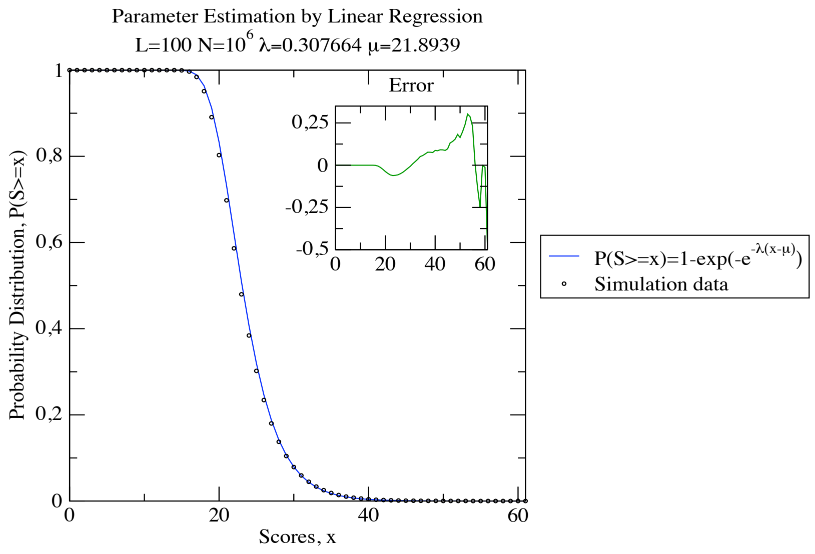


FIGURE 3.7. Probability distribution plot for sequences of length $L = 100$.

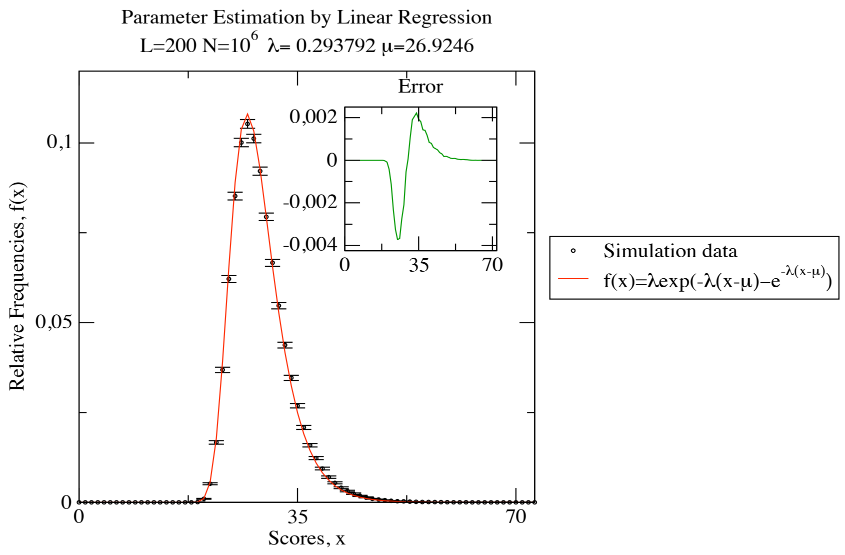


FIGURE 3.8. Probability density plot of simulated and estimated data for sequences of length $L = 200$.

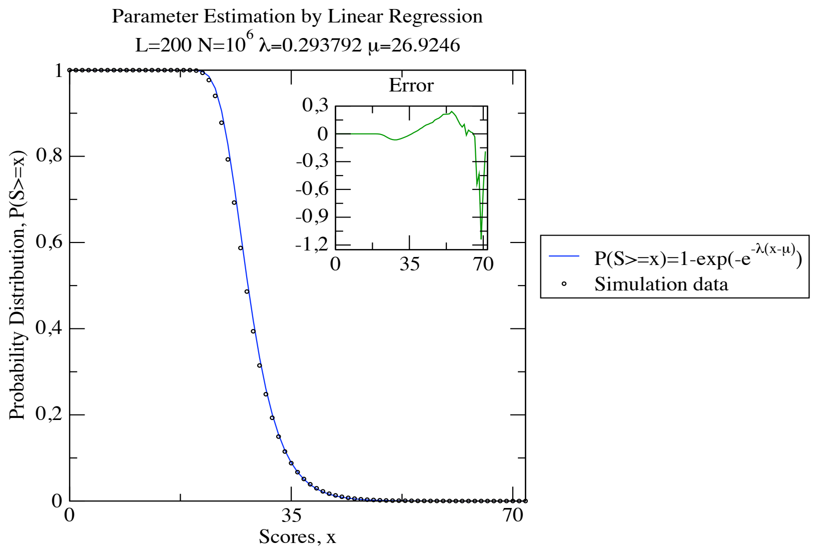


FIGURE 3.9. Probability distribution plot for sequences of length $L = 200$.

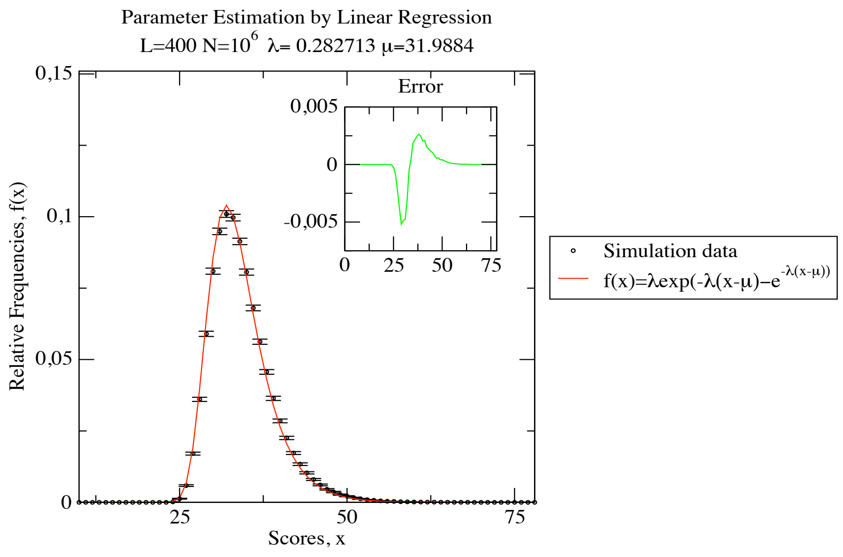


FIGURE 3.10. Probability density plot of simulated and estimated data for sequences of length $L = 400$.

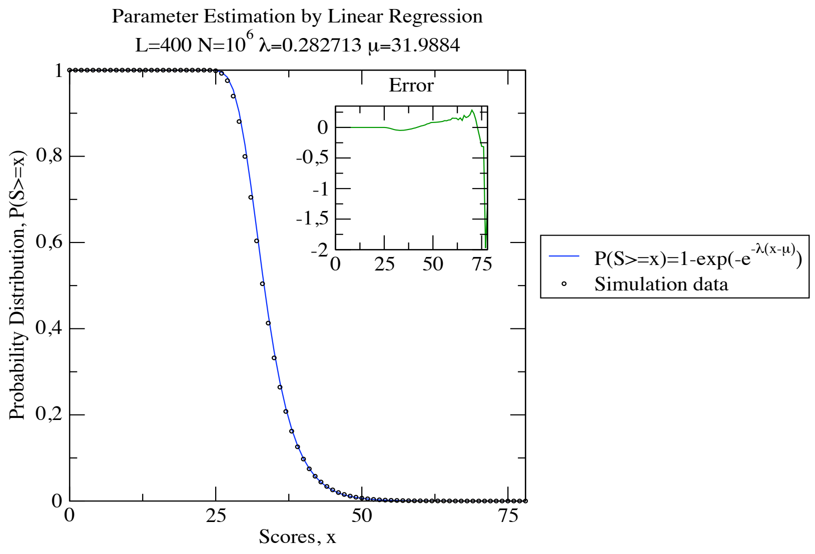


FIGURE 3.11. Probability distribution plot for sequences of length $L = 400$.

Chapter 4

Conclusions

Recent discoveries in molecular biology revealed new facts about the structure of biological sequences allowing biological research to become more theoretical. Namely, there is an opportunity to study the structure and the function of biological sequences via sequence comparison methods. Here we studied sequence alignment, a powerful method for biosequence comparison. This method effectively combines knowledge from biology, mathematics, computer science, and statistics. We illustrated that dynamic programming algorithms are effective computational tools for sequence alignment, and statistical methods are essential in determining the significance of alignment scores.

The extreme value theory proved to be very useful in the statistics of sequence alignment. It plays the central role in the explicit formula for distribution of ungapped local alignment scores. It also gives a good approximation in the case of gapped local alignments. To illustrate this fact in more detail we performed simulations and estimated parameters of extreme value type-I distribution for gapped local alignment scores by linear regression, the most simple and straightforward method known. The advantage of this method is a combination of simplicity and very good estimates in the regions of distribution, where probabilities are relatively large. However, this method it is not effective if we are specifically interested in very small probabilities, which is often the case in biological research. Thus, for estimation of small probabilities more advanced methods have to be used [25].

Although many results have already been obtained in the area of assessment of statistical significance of biosequence analysis, many important questions remain open. For example, explicit theory for distribution of gapped local alignment scores and general efficient methods for global alignments are still absent [32].

Finally, we must remember that besides providing interesting mathematical and computational research problems, biological sequence comparison helps answering very important biological questions such as gene regulation in living organisms, evolution, etc. With a deeper understanding of biological sequences and the roles they play in living organisms, scientists will be better positioned to develop tools and technologies that improve human health [21].

Bibliography

1. S. F. Altschul, *Gap costs for multiple sequence alignment*, J. Theor. Biol. **138** (1989Jun), 297–309.

2. S. F. Altschul, R. Bundschuh, R. Olsen, and T. Hwa, *The estimation of statistical parameters for local alignment score distributions*, Nucleic Acids Res. **29** (2001Jan), 351–361.

3. S. F. Altschul and W. Gish, *Local alignment statistics*, Meth. Enzymol. **266** (1996), 460–480.

4. S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman, *Basic local alignment search tool*, J. Mol. Biol. **215** (1990Oct), 403–410.

5. R. Arratia, L. Gordon, and M.S. Waterman, *The Erdös-Renýi law in distribution, for coin tossing and sequence matching*, Ann. Stat. **18** (1990), 539–570.

6. M. P. Ball, *Dna chemical structure — Wikimedia Commons*, http://commons.wikimedia.org/wiki/File:DNA_chemical_structure.svg. [Online; accessed April-2010].

7. ______, *Genetic code — Wikimedia Commons*, http://commons.wikimedia.org/wiki/File:Genetic_code.svg, 2006. [Online; accessed April-2010].

8. L. J. Beamer and C. O. Pabo, *Refined 1.8 A crystal structure of the lambda repressor-operator complex*, J. Mol. Biol. **227** (1992Sep), 177–196.

9. S. A. Benner, M. A. Cohen, and G. H. Gonnet, *Empirical and structural models for insertions and deletions in the divergent evolution of proteins*, J. Mol. Biol. **229** (1993Feb), 1065–1082.

10. K. M. Chao and L. Zhang, *Sequence Comparison. Theory and Methods*, Springer, 2009.

11. E. Chargaff, S. Zamenhof, and C. Green, *Composition of human desoxypentose nucleic acid*, Nature **165** (1950May), 756–757.

12. J. E. Cohen, *Mathematics is biology's next microscope, only better; biology is mathematics' next physics, only better*, PLoS Biol. **2** (2004Dec), 439.

13. T. Cormen, C. Leiserson, R. Rivest, and C. Stein, *Introduction to Algorithms*, 3rd ed., MIT Press, 2009.

14. F. Crick, *Life Itself. Its Origin and Nature*, Simon and Shuster, 1995.

15. A. Dembo, S. Karlin, and O. Zeitouni, *Critical phenomena for sequence matching with scoring*, Ann. Probab. **22** (1994), 1993–2021.

16. ______, *Limit distribution of maximal non-aligned two sequence segmental score*, Ann. Probab. **22** (1994), 2022–2039.

17. R. F. Doolittle, M. W. Hunkapiller, L. E. Hood, S. G. Devare, K. C. Robbins, S. A. Aaronson, and H. N. Antoniades, *Simian sarcoma virus onc gene, v-sis, is derived from the gene (or genes) encoding a platelet-derived growth factor*, Science **221** (1983Jul), 275–277.

18. R. Durbin, S. Eddy, A. Krogh, and G. Mitchison, *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*, Cambridge University Press, 1999.

19. S. R. Eddy, *Where did the BLOSUM62 alignment score matrix come from?*, Nat. Biotechnol. **22** (2004Aug), 1035–1036.

20. Feller, *An Introduction to Probability Theory and Its Applications*, 2nd ed., Vol. 2, John Wiley and Sons, New York, 1987.

21. Progress from The National Institute of General Medical Sciences, *Genes — what we knew, know and hope to learn*, http://www.nigms.nih.gov/Publications/FactSheet_Genes.htm. [Online; accessed April-2010].

22. O. Gotoh, *An improved algorithm for matching biological sequences*, J. Mol. Biol. **162** (1982Dec), 705–708.

23. G. Grimmett and D. Stirzaker, *Probability and Random Processes*, 3rd ed., Oxford University Press Inc., New York, 2001.

24. Röst Hannes, *Central dogma of molecular biology — Wikimedia Commons*, http://commons.wikimedia.org/wiki/File:BLOSUM62.gif, 2008. [Online; accessed April-2010].

25. A. K. Hartmann, *Sampling rare events: statistics of local sequence alignments*, Phys Rev E Stat Nonlin Soft Matter Phys **65** (2002May), 056102.

26. S. Henikoff and J. G. Henikoff, *Automated assembly of protein blocks for database searching*, Nucleic Acids Res. **19** (1991Dec), 6565–6572.

27. ______, *Amino acid substitution matrices from protein blocks*, Proc. Natl. Acad. Sci. U.S.A. **89** (1992Nov), 10915–10919.

28. N. Hulo, A. Bairoch, V. Bulliard, L. Cerutti, E. De Castro, P. S. Langendijk-Genevaux, M. Pagni, and C. J. Sigrist, *The PROSITE database*, Nucleic Acids Res. **34** (2006Jan), 227–230.

29. S. Karlin and S. F. Altschul, *Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes*, Proc. Natl. Acad. Sci. U.S.A. **87** (1990Mar), 2264–2268.

30. ______, *Applications and statistics for multiple high-scoring segments in molecular sequences*, Proc. Natl. Acad. Sci. U.S.A. **90** (1993Jun), 5873–5877.

31. S. Karlin and A. Dembo, *Limit distributions of maximal segmental score among markov-dependent partial sums*, Adv. Apl. Prob. **24** (1992), 113–140.

32. A. Yu. Mitrophanov and M. Borodovsky, *Statistical significance in biological sequence analysis*, Briefings in Bioinformatics (2006), 2–24.

33. R. Mott, *Accurate formula for P-values of gapped local sequence and profile alignments*, J. Mol. Biol. **300** (2000Jul), 649–659.

34. S. B. Needleman and C. D. Wunsch, *A general method applicable to the search for similarities in the amino acid sequence of two proteins*, J. Mol. Biol. **48** (1970Mar), 443–453.

35. United States Department of Energy, *Dna complementarity — Wikimedia Commons*, http://commons.wikimedia.org/wiki/File:Dna-split.png. [Online; accessed April-2010].

36. L. Pachter and B. Sturmfels, *Algebraic Statistics for Computational Biology*, Cambridge University Press, 2005.

37. Y. Park, S. Sheetlin, and J. Spouge, *Accelerated convergence and robust asymptotic regression of the Gumbel scale parameter for gapped sequence alignment*, J. Phys. A: Math. Gen. **38** (2005), 97–108.

38. W. R. Pearson, *Empirical statistical estimates for sequence similarity searches*, J. Mol. Biol. **276** (1998Feb), 71–84.

39. R. Rifkind and C. Rifkind, *Naturally Obsessed: The Making of A Scientist*, http://naturallyobsessed.com/.

40. A. B. Robinson and L. R. Robinson, *Distribution of glutamine and asparagine residues and their near neighbors in peptides and proteins*, Proc. Natl. Acad. Sci. U.S.A. **88** (1991Oct), 8880–8884.

41. T. F. Smith and M. S. Waterman, *Identification of common molecular subsequences*, J. Mol. Biol. **147** (1981Mar), 195–197.

42. T. F. Smith, M. S. Waterman, and C. Burks, *The statistical distribution of nucleic acid similarities*, Nucleic Acids Res. **13** (1985Jan), 645–656.

43. F. Spitzer, *Principles of Random Walk*, 2nd ed., Springer Verlag, New York, Berlin, Heidenberg, 2001.

44. M. S. Waterman and M. Vingron, *Rapid and accurate estimates of statistical significance for sequence data base searches*, Proc. Natl. Acad. Sci. U.S.A. **91** (1994May), 4625–4628.

45. R. Wheeler, *The lambda repressor-operator complex 1LMB*, `http://en.wikipedia.org/wiki/User:Zephyris`. [Online; accessed April-2010].

46. Wikipedia, *Carbon — Wikipedia, the free encyclopedia*, `http://en.wikipedia.org/wiki/Carbon`. [Online; accessed April-2010].

47. ———, *Hypothetical types of biochemistry — Wikipedia, the free encyclopedia*, `http://en.wikipedia.org/wiki/Hypothetical_types_of_biochemistry`. [Online; accessed April-2010].

48. ———, *Neznaika — Wikipedia, the free encyclopedia*, `http://en.wikipedia.org/wiki/Neznaika`. [Online; accessed April-2010].

49. M. A. Zachariah, G. E. Crooks, S. R. Holbrook, and S. E. Brenner, *A generalized affine gap model significantly improves protein sequence alignment accuracy*, Proteins **58** (2005Feb), 329–338.

Appendix A

C++ Code

```
/*********************************************************************************/
/******************************** SIMULATIONS ************************************/
/** Estimation of parameters Lambda and Mu of extreme value type-I distribution   **/
/*********************************************************************************/
/** CONSTANTS:                                                                   **/
/**           BLOSUM62: protein substitution matrix BLOSUM62                     **/
/**             RRfreq: Robinson and Robinson cumulative frequencies             **/
/**                  C: [-C,C] Normal distribution confidence interval           **/
/**              amino: amino acid ordering                                      **/
/*********************************************************************************/
/** PARAMETERS:                                                                  **/
/**                  L: sequence length;                                         **/
/**                  N: number of sequence pairs;                                **/
/**                  O: additional penalty for opening a gap;                    **/
/**                  e: penalty for continuing a gap;                            **/
/**           fileNAME: name of the experiment;                                  **/
/*********************************************************************************/
/** RETURNS:                                                                     **/
/**             Lambda: scaling parameter;                                       **/
/**                 Mu: location parameter;                                      **/
/**               Corr: correlation coefficient;                                 **/
/**          diffclock: execution time;                                          **/
/**   [indmin, indmax]: score range;                                             **/
/**                  i: score                                                    **/
/**           Count[i]: number of counts (or Simulated P(S=i)*N)                 **/
/**       Error_bar[i]: error bars for score counts                              **/
/**            Prob[i]: Simulated P(S<=i)                                        **/
/**          Gumbel[i]: Gumbel P(S=i) with parameters Lambda and Mu              **/
/**     Gumbel_Prob[i]: Gumbel P(S<=i) with parameters Lambda and Mu             **/
/**           Error[i]: Simulated P(S=i) - Gumbel P(S=i)                         **/
/**      Error_Prob[i]: (Simulated P(S<=i) - Gumbel P(S<=i))/Simulated P(S<=i)   **/
/*********************************************************************************/
/** OUTPUT FILES:                                                                **/
/** fileNAME_Sim_Dens.dat     i  Count[i]  Error_bar[i]                          **/
/** fileNAME_Sim_Prob.dat     i  Prob[i]                                         **/
/** fileNAME_Gumb_Dens.dat    i  Gumbel_Dens[i]                                  **/
/** fileNAME_Gumb_Prob.dat    i  Gumbel_Prob[i]                                  **/
/** fileNAME_Err_Dens.dat     i  Error[i]                                        **/
/** fileNAME_Err_Prob.dat     i  Error_Prob[i]                                   **/
/** fileNAME_RESULTS.dat      i  P(S=i)  P(S<=i) log(-log(P(S<=y)))              **/
/**                           BLOSUM62, O, e, L, N, minscore, maxscore,          **/
/**                           estimated Lambda and Mu,                           **/
/**                           correlation coefficient.                           **/
/******************************** THE END :) *************************************/
/*********************************************************************************/
#include <stdlib.h>
```

```
#include <iostream>
#include <fstream>
#include <sstream>
#include <string>
#include <math.h>
#include <ctime>
#include <gsl/gsl_rng.h>
using namespace std;


                                                                       /* FUNCTIONS */
double max(double array[], int l);              /* Finds maximum value in the array */
int first_ind(double array[], int l);                      /* Computes minimum score */
int last_ind(double array[], int l);                       /* Computes maximum score */
double diffclock(clock_t clock1,clock_t clock2);         /* Computes execution time */

int main(){
                                                              /* DEFAULT PARAMETERS */
        int           L=40,                                      /* Sequence length */
                      N=1000,                                    /* Number of pairs */
                      e=-1,                                /* Gap extension penalty */
                      O=-11;                     /* Additional gap opening penalty */
        string  fileNAME="Experiment";                    /* Name of the experiment */

                                                                       /* CONSTANTS */
        const char amino[]=
   /* 1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20 */
    {'A','R','N','D','C','Q','E','G','H','I','L','K','M','F','P','S','T','W','Y','V'};

        const int BLOSUM62[]=
                        /* A  R  N  D  C  Q  E  G  H  I  L  K  M  F  P  S  T  W  Y  V */
                /* A */   {4,-1,-2,-2, 0,-1,-1, 0,-2,-1,-1,-1,-1,-2,-1, 1, 0,-3,-2, 0,
                /* R */   -1, 5, 0,-2,-3, 1, 0,-2, 0,-3,-2, 2,-1,-3,-2,-1,-1,-3,-2,-3,
                /* N */   -2, 0, 6, 1,-3, 0, 0, 0, 1,-3,-3, 0,-2,-3,-2, 1, 0,-4,-3,-3,
                /* D */   -2,-2, 1, 6,-3, 0, 2,-1,-1,-3,-4,-1,-3,-3,-1, 0,-1,-4,-3,-3,
                /* C */    0,-3,-3,-3, 9,-3,-4,-3,-3,-1,-1,-3,-1,-2,-3,-1,-1,-2,-2,-1,
                /* Q */   -1, 1, 0, 0,-3, 5, 2,-2, 0,-3,-2, 1, 0,-3,-1, 0,-1,-2,-1,-2,
                /* E */   -1, 0, 0, 2,-4, 2, 5,-2, 0,-3,-3, 1,-2,-3,-1, 0,-1,-3,-2,-2,
                /* G */    0,-2, 0,-1,-3,-2,-2, 6,-2,-4,-4,-2,-3,-3,-2, 0,-2,-2,-3,-3,
                /* H */   -2, 0, 1,-1,-3, 0, 0,-2, 8,-3,-3,-1,-2,-1,-2,-1,-2,-2, 2,-3,
                /* I */   -1,-3,-3,-3,-1,-3,-3,-4,-3, 4, 2,-3, 1, 0,-3,-2,-1,-3,-1, 3,
                /* L */   -1,-2,-3,-4,-1,-2,-3,-4,-3, 2, 4,-2, 2, 0,-3,-2,-1,-2,-1, 1,
                /* K */   -1, 2, 0,-1,-3, 1, 1,-2,-1,-3,-2, 5,-1,-3,-1, 0,-1,-3,-2,-2,
                /* M */   -1,-1,-2,-3,-1, 0,-2,-3,-2, 1, 2,-1, 5, 0,-2,-1,-1,-1,-1, 1,
                /* F */   -2,-3,-3,-3,-2,-3,-3,-3,-1, 0, 0,-3, 0, 6,-4,-2,-2, 1, 3,-1,
                /* P */   -1,-2,-2,-1,-3,-1,-1,-2,-2,-3,-3,-1,-2,-4, 7,-1,-1,-4,-3,-2,
                /* S */    1,-1, 1, 0,-1, 0, 0, 0,-1,-2,-2, 0,-1,-2,-1, 4, 1,-3,-2,-2,
                /* T */    0,-1, 0,-1,-1,-1,-1,-2,-2,-1,-1,-1,-1,-2,-1, 1, 5,-2,-2, 0,
                /* W */   -3,-3,-4,-4,-2,-2,-3,-2,-2,-3,-2,-3,-1, 1,-4,-3,-2,11, 2,-3,
                /* Y */   -2,-2,-2,-3,-2,-1,-2,-3, 2,-1,-1,-2,-1, 3,-3,-2,-2, 2, 7,-1,
                /* V */    0,-3,-3,-3,-1,-2,-2,-3,-3, 3, 1,-2, 1,-1,-2,-2, 0,-3,-1, 4},

                  RRfreq[]=
                                       /* A */                                     { 31155,
                                       /* A+R */                                     58260,
                                       /* A+R+N */                                   78472,
                                       /* A+R+N+D */                                102633,
                                       /* A+R+N+D+C */                              111302,
                                       /* A+R+N+D+C+Q */                           130510,
                                       /* A+R+N+D+C+Q+E */                         158864,
                                       /* A+R+N+D+C+Q+E+G */                       192093,
                                       /* A+R+N+D+C+Q+E+G+H */                     201999,
                                       /* A+R+N+D+C+Q+E+G+H+I */                   225160,
                                       /* A+R+N+D+C+Q+E+G+H+I+L */                 265785,
                                       /* A+R+N+D+C+Q+E+G+H+I+L+K */               291657,
                                       /* A+R+N+D+C+Q+E+G+H+I+L+K+M */             301758,
                                       /* A+R+N+D+C+Q+E+G+H+I+L+K+M+F */           319125,
                                       /* A+R+N+D+C+Q+E+G+H+I+L+K+M+F+P */         342560,
                                       /* A+R+N+D+C+Q+E+G+H+I+L+K+M+F+P+S */       374630,
                                       /* A+R+N+D+C+Q+E+G+H+I+L+K+M+F+P+S+T */     400941,
                                       /* A+R+N+D+C+Q+E+G+H+I+L+K+M+F+P+S+T+W */   406931,
                                       /* A+R+N+D+C+Q+E+G+H+I+L+K+M+F+P+S+T+W+Y */ 421419,
                                       /* A+R+N+D+C+Q+E+G+H+I+L+K+M+F+P+S+T+W+Y+V */450431},

   Num_score=100;
   const double C=1.96;                 /* Normal distribution 95% confidence interval*/
   int          i                                                            /* Score */
                j,q,
                 ind,
                 MaxScore,                /* Maximum score in local pairwise alignment */
                 indmin,indmax;
   double       rnd,
                 Count[Num_score],                    /* Simulated score frequencies */
                 Prob[Num_score],                  /* Simulated cumulative probability */
                 Prob1[Num_score],
                 Error_bar[Num_score],          /* Error bars for counting scores (5%) */
                 Sx, Sy, Sxy, Sx2,                                /* Regression sums */
                 Sy2, Sn,
                 Corr,                                     /* Correlation coefficient */
                 Lambda, Mu,                                  /* Estimated parameters */
                 Gumbel[Num_score],                /* Estimated probability density */
                 Gumbel_Prob[Num_score],         /* Estimated cumulative probability */
                 Error[Num_score],                                           /* Errors */
                 Error_Prob[Num_score];

   string       file1,file2,file3,
                 file4,file5,file6,
                 file7;

        clock_t before=clock();
        cout<< "*****************************************************************"<<endl;
        cout<< "**  WELCOME TO THE GAPPED LOCAL ALIGNMENT STATISTICS SIMULATIONS! **"<<endl;
        cout<< "*****************************************************************"<<endl;
        cout<< "****************** DEFAULT PARAMETERS: ***************************"<<endl;
        cout<< "Substitution matrix: BLOSUM62"<<endl;
        cout<< "Affine gap penalty: O = "<<O<<" and e = "<<e<<endl;
        cout<< "Sequence length: L = "<<L<<endl;
        cout<< "Number of sequence pairs : N = "<<N<<endl;
        cout<< "Name of the experiment : "<<fileNAME<<endl;
        cout<< "*****************************************************************"<<endl;
    char Ans;
    int  Par;
        cout<< "Would you like to reset parameters (Y/N)? ";
        cin >> Ans;
        if(Ans == 'Y' || Ans == 'y' ){
             cout << "Sequence length, L = ";
```

```
            cin >> Par;
            L=Par;
        cout << "Number of pairs, N = ";
            cin >> Par;
            N=Par;
        cout << "Penalty for opening a gap, O = ";
            cin >> Par;
            O=Par;
        cout << "Penalty for extending a gap, e = ";
            cin >> Par;
            e=Par;
        cout << "Name of the experiment: ";
            cin >> fileNAME;
    };
char     Seq[2*L];                  /* Amino acid representation of pair of sequences */
int      Seq_Num[2*L];                 /* Integer representation of pair of sequences */

double
            S[(L+1)*(L+1)],         /* Substitution, Deletion, and Insertion matrices */
            D[(L+1)*(L+1)],
            I[(L+1)*(L+1)],
            rec_S[4],                                            /* Recursion arrays */
            rec_D[2],
            rec_I[2],

    gsl_rng *rng;                /* Initialization of pseudorandom number generator */
    srand((unsigned)time(0));                  /* We used mt19937 from GSL library  */
    rng=gsl_rng_alloc(gsl_rng_mt19937);
    gsl_rng_set(rng, rand());

    for (i=0;i<Num_score;i++){
            Count[i]=0;
    }
    for (q=1;q<=N;q++){
        for(i=0;i<2*L;i++){        /* Generating a pair of random protein sequences*/
            rnd=gsl_rng_uniform(rng)*450431;
           if (rnd<=RRfreq[0]){Seq[i]=amino[0];Seq_Num[i]=0;}
                            for (j=1;j<=20;j++){
                                    if (rnd<=RRfreq[j]&&rnd>=RRfreq[j-1])
                                    {Seq[i]=amino[j]; Seq_Num[i]=j;}
                            }
    }                 /* Aligning sequences by SMITH-WATERMAN ALGORITHM (affine gap) */
                    for(i=0;i<=L-1;i++){                          /* Initialization  */
                            for (j=0;j<=L-1;j++){
                                    S[i*L+j]=0;
                                    D[i*L+j]=0;
                                    I[i*L+j]=0;
                            }
                    }
                    for (i=0;i<=L;i++){
                                    D[i*L]=-100;
                                    I[i*L]=-100;
                                    S[i*L]=0;
                    }
                    for (j=0;j<=L;j++){
                                    I[j]=-100;
                                    D[j]=-100;
                                    S[j]=0;
```

```
                for (i=1;i<=L;i++){                                       /* Recursion */
                        for (j=1;j<=L;j++){
                                ind=(Seq_Num[i-1])*20+Seq_Num[L+j-1];
                                rec_S[0]=S[(i-1)*L+j-1]+BLOSUM62[ind];
                                rec_S[1]=D[(i-1)*L+j-1]+BLOSUM62[ind];
                                rec_S[2]=I[(i-1)*L+j-1]+BLOSUM62[ind];
                                rec_S[3]=0;
                                S[i*L+j]=max(rec_S,4);
                                rec_D[0]=S[(i-1)*L+j]+O+e;
                                rec_D[1]=D[(i-1)*L+j]+e;
                                D[i*L+j]=max(rec_D,2);
                                rec_I[0]=S[i*L+j-1]+O+e;
                                rec_I[1]=I[i*L+j-1]+e;
                                I[i*L+j]=max(rec_I,2);
                        }
                }

                MaxScore=0;                             /* Finding maximum score */
                for (i=1;i<=L;i++){
                        for (j=1;j<=L;j++){
                                if(S[L*i+j]>MaxScore){
                                        MaxScore = S[L*i+j];
                                }
                        }
                }
                if (MaxScore>=0&&MaxScore<Num_score){    /* Counting maximum score */
                    Count[MaxScore]++;
                }
}
Prob[0]=0;               /* Calculating frequencies, error bars and probabilities */
Prob1[0]=log(-1*(log(Prob[0])));
Error_bar[0]=(sqrt(Count[0]*(1-(Count[0]/N)))*C*2)/N;
for(i=1;i<Num_score;i++){
         Error_bar[i]=(sqrt(Count[i]*(1-(Count[i]/N)))*C*2)/N;
         Prob[i]=Prob[i-1]+Count[i]/N;
         Prob1[i]=log(double((-1)*log(Prob[i])));
}                                                           /* Organizing output */
file1=fileNAME+"_Sim_Dens.dat";
file2=fileNAME+"_Sim_Prob.dat";
file3=fileNAME+"_Gumb_Dens.dat";
file4=fileNAME+"_Gumb_Prob.dat";
file5=fileNAME+"_Err_Dens.dat";
file6=fileNAME+"_Err_Prob.dat";
file7=fileNAME+"_RESULTS.dat";
ofstream fout1(file1.c_str(), ios::ate);
ofstream fout2(file2.c_str(), ios::ate);
ofstream fout3(file3.c_str(), ios::ate);
ofstream fout4(file4.c_str(), ios::ate);
ofstream fout5(file5.c_str(), ios::ate);
ofstream fout6(file6.c_str(), ios::ate);
ofstream fout7(file7.c_str(), ios::ate);
cout<< "*****************************************************************"<<endl;
cout<< "******************** DISTRIBUTION OF SCORES **********************"<<endl;
cout<< "*****************************************************************"<<endl;
cout<<"y"<<'\t'<<"P(S=y)"<<'\t'<<"P(S<=y)"<<'\t'<<"log(-log(P(S<=y)))"<<endl;
fout7<< "*****************************************************************"<<endl;
fout7<< "******************* DISTRIBUTION OF SCORES ***********************"<<endl;
```

```
  fout7<< "*******************************************************************"<<endl;
  fout7<<"y"<<'\t'<<"P(S=y)"<<'\t'<<"P(S<y)"<<'\t'<<"log(-log(P(S<y)))"<<endl;
  indmin=first_ind(Count, Num_score);
  indmax=last_ind(Prob, Num_score);
  for(i=indmin;i<=indmax;i++){
          cout<<i<<'\t'<<Count[i]/(N)<<'\t'<<Prob[i]<<'\t'<<Prob1[i]<<endl;
          fout7<<i<<'\t'<<Count[i]/(N)<<'\t'<<Prob[i]<<'\t'<<Prob1[i]<<endl;
          fout1<<i<<" "<<Count[i]/(N)<<" "<<Error_bar[i]<<endl;
          fout2<<i<<" "<<1-Prob[i]<<endl;
  }
  Sy=Prob1[indmin];                                                /* Regression */
  Sx=indmin;
  Sxy=Sx*Sy;
  Sx2=Sx*Sx;
  Sy2=Sy*Sy;
  for(i=indmin+1;i<indmax;i++){
          Sy=Sy+Prob1[i];
          Sx=Sx+i;
          Sxy=Sxy+i*Prob1[i];
          Sx2=Sx2+i*i;
          Sy2=Sy2+Prob1[i]*Prob1[i];
  }
  Sn=indmax-indmin;
  Lambda=-1*(Sn*Sxy-Sx*Sy)/(Sn*Sx2-Sx*Sx);                  /* Lambda Estimation */
  Mu=(Sy+Lambda*Sx)/(Sn*Lambda);                                /* Mu Estimation */
  Corr=(Sn*Sxy-Sx*Sy)/(sqrt(Sn*Sx2-Sx*Sx)*sqrt(Sn*Sy2-Sy*Sy));
  cout<< "**************************** PARAMETERS ******************************"<<endl;
  cout<<"Substitution matrix:     BLOSUM62"<<endl;
  cout<<"Affine gap penalty:      O = "<<O<<" and e = "<<e<<endl;
  cout<<"Sequence length:         "<<L<<endl;
  cout<<"Number of pairs:         "<<N<<endl;
  cout<<"Minimum score:           "<<indmin<<endl;
  cout<<"Maxumum score:           "<<indmax<<endl;
  cout<<"Regression Lambda:       "<<Lambda<<endl;
  cout<<"Regression Mu:           "<<Mu<<endl;
  cout<<"Correlation coefficient  "<<Corr<<endl;
  cout<< "**************************** THE END :) *******************************"<<endl;
  fout7<< "**************************** PARAMETERS ******************************"<<endl;
  fout7<<"Substitution matrix:     BLOSUM62"<<endl;
  fout7<< "Affine gap penalty:     O = "<<O<<" and e = "<<e<<endl;
  fout7<<"Sequence Length:         "<<L<<endl;
  fout7<<"Number of pairs:         "<<N<<endl;
  fout7<<"Minimum score:           "<<indmin<<endl;
  fout7<<"Maximum score:           "<<indmax<<endl;
  fout7<<"Regression Lambda:       "<<Lambda<<endl;
  fout7<<"Regression Mu:           "<<Mu<<endl;
  fout7<<"Correlation coefficient: "<<Corr<<endl;
  fout7<< "**************************** THE END :) ******************************"<<endl;
/* Computing extreme value type-I density and distribution with parameter Lambda and Mu */
                              /* Computing errors between simulated and estimated data */
  for (i=0;i<Num_score;i++){
          Gumbel[i]=Lambda*exp(-Lambda*(i-Mu))*exp(-exp(-Lambda*(i-Mu)));
          fout3<<i<<" "<<Gumbel[i]<<endl;
          Gumbel_Prob[i]=1-exp(-exp(-Lambda*(i-Mu)));
          fout4<<i<<" "<<Gumbel_Prob[i]<<endl;
          Error[i]=Count[i]/(N)-Gumbel[i];
          fout5<<i<<" "<<Error[i]<<endl;
          Error_Prob[i]=(1-Prob[i]-Gumbel_Prob[i])/(1-Prob[i]);
```

```
                fout6<<i<<" "<<Error_Prob[i]<<endl;
        }
        clock_t after=clock();
        cout<<"Execution Time:"<<'\t'<<diffclock(after,before)<<" sec"<<endl;
        fout7<<"Execution Time:"<<'\t'<<diffclock(after,before)<<" sec"<<endl;
        fout1.close();
        fout2.close();
        fout3.close();
        fout4.close();
        fout5.close();
        fout6.close();
        fout7.close();
        return 0;
}
int first_ind(double array[], int l){
        int ind=0;
        for (int i=0; i<l;i++){
        if (array[i]>0){ind=i;break;}
        }
        return ind;
}
int last_ind(double array[], int l){
        int ind=0;
        for (int i=0;i<l;i++){
        if(array[l-i-1]>0) {ind=l-i-1; break;}
        }
        return ind;
}
double max(double array[], int l){
double max_val=array[0];
         for (int i=1;i<l;i++){
                if(array[i]>max_val){
                       max_val=array[i];
               }
         }
         return max_val;
}
double diffclock(clock_t clock1,clock_t clock2)
{
        double diffticks=clock1-clock2;
        double diffms=(diffticks)/CLOCKS_PER_SEC;
        return diffms;
}
```