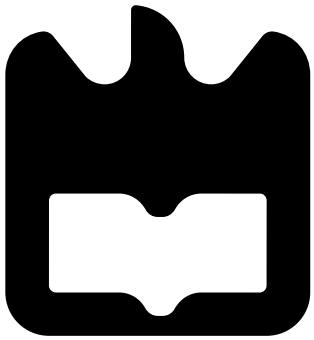
Universidade de Aveiro Departamento de Electrónica, Telecomunicações e Informática,
2017

Manuel Augusto Ribeiro Gaspar

Sistema de pesquisa automática de sequências de ADN aproximadas e não contíguas

Automatic system for approximate and noncontiguous DNA sequences search



Universidade de Aveiro Departamento de Electrónica, Telecomunicações e Informática,
2017

Manuel Augusto Ribeiro Gaspar

Sistema de pesquisa automática de sequências de ADN aproximadas e não contíguas

Automatic system for approximate and noncontiguous DNA sequences search

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Eletrónica e Telecomunicações, realizada sob a orientação científica do Doutor Armando José Formoso de Pinho, Professor associado com agregação do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro, e do Doutor Diogo Rodrigo Marques Pratas, Bolseiro de Pós-Doutoramento da Universidade de Aveiro.

o júri / the jury

presidente / president

Professora Doutora Ana Maria Perfeito Tomé
Professora Associada, Universidade de Aveiro

vogais / examiners committee

Professora Doutora Bernardete Martins Ribeiro
Professora Associada com Agregação, Universidade de Coimbra - Faculdade de Ciências e Tecnologia

Doutor Diogo Rodrigo Marques Pratas
Bolseiro de Pós-Doutoramento, Universidade de Aveiro

agradecimentos / acknowledgements

Em primeiro lugar devo agradecer ao meu orientador, Doutor Armando José Formoso de Pinho, e ao meu co-orientador, Doutor Diogo Rodrigo Marques Pratas, pelo apoio dado e pelo interesse que demonstraram ao longo da realização deste trabalho.

Aos meus pais e irmãos, pelo apoio dado e pela confiança que depositaram em mim e na minha capacidade de atingir este objectivo. À restante família, pelo interesse na minha felicidade e bem-estar.

A todos os amigos, sejam os novos amigos que fiz durante este percurso, ou os de longa data, que trago comigo há tanto tempo, por todos os momentos de descontração e boa disposição.

A todos os docentes que, de uma forma ou outra, terão contribuído para o meu desenvolvimento como pessoa.

Este trabalho marca o fim de uma etapa e o início da próxima, e, para além de meu, é também um pouco de todos vocês. Obrigado por tudo.

Palavras Chave

ADN, gene, cromossoma, genoma, região específica humana, pesquisa de sequências, compressão de dados, similaridade, informação, modelos de contexto finito, estatística.

Resumo

A capacidade de efectuar pesquisas de sequências de ADN similares a outras contidas numa sequência maior, tal como um cromossoma, tem um papel muito importante no estudo de organismos e na possível ligação entre espécies diferentes.

Apesar da existência de várias técnicas e algoritmos, criados com o intuito de realizar pesquisas de sequência, este problema ainda está aberto ao desenvolvimento de novas ferramentas que possibilitem melhorias em relação a ferramentas já existentes.

Esta tese apresenta uma solução para pesquisa de sequências, baseada em compressão de dados, ou, mais especificamente, em modelos de contexto finito, obtendo uma medida de similaridade entre uma referência e um alvo. O método usa uma abordagem com base em modelos de contexto finito para obtenção de um modelo estatístico da sequência de referência e obtenção do número estimado de bits necessários para codificação da sequência alvo, utilizando o modelo da referência.

Ao longo deste trabalho, estudámos o método descrito acima, utilizando, inicialmente, condições controladas, e, por fim, fazendo um estudo de regiões de ADN do genoma humano moderno, que não se encontram em ADN ancestral (ou se encontram com elevado grau de dissimilaridade).

Keywords DNA, gene, chromosome, genome, human specific regions, sequence search, data compression, similarity, information, finite-context models, statistics.

Abstract

The ability to search similar DNA sequences with relation to a larger sequence, such as a chromosome, has a really important role in the study of organisms and the possible connection between different species.

Even though several techniques and algorithms, created with the goal of performing sequence searches, already exist, this problem is still open to the development of new tools that exhibit improvements over currently existent tools.

This thesis proposes a solution for sequence search, based on data compression, or, specifically, finite-context models, by obtaining a measure of similarity between a reference and a target. The method uses an approach based on finite-context models for the creation of a statistical model of the reference sequence and obtaining the estimated number of bits necessary for the codification of the target sequence, using the reference model.

In this work we studied the above described method, using, initially, controlled conditions, and, finally, conducting a study on DNA regions, belonging to the modern human genome, that can not be found in ancient DNA (or can only be found with high dissimilarity rate).

CONTENTS

Contents i

List of Figures iii

List of Tables v

Glossary vii

1 Introduction 1
- 1.1 Main contributions . . . 2
- 1.2 Thesis organization . . . 2

2 Biological Background 3
- 2.1 DNA Overview . . . 3
  - 2.1.1 Inverted complements . . . 6
  - 2.1.2 Three-base periodicity . . . 6

3 Data Compression 9
- 3.1 Information Theory . . . 9
- 3.2 Compression techniques . . . 11
  - 3.2.1 Variable-length coding . . . 11
  - 3.2.2 Dictionary-based compression . . . 12
  - 3.2.3 Arithmetic coding . . . 13
    - Encoding . . . 14
    - Decoding . . . 14
- 3.3 Finite-context model (FCM) . . . 14
- 3.4 Genomic sequence compression . . . 17
  - 3.4.1 Individual compression . . . 17

3.4.2 Reference compression 18
3.5 Kolmogorov Complexity: the lower bound of compression 19

4 Compression based sequence search **21**
4.1 The method 22
4.1.1 Three-state model 22
4.1.2 Inverted complements 22
4.1.3 Data models application 24
4.1.4 Auxiliary tools 27
Filtering 27
4.2 Implementation 27

5 Results **31**
5.1 Synthetic Sequences 31
5.1.1 Results 33
5.2 Real Sequences 35
5.2.1 Exact Gene Search 35
5.2.2 Homolog genes 38
Results for gene MYO3A 39
Results for gene SPATA45 40
Results for gene SYT1 41
5.3 Application to ancient DNA 44
5.3.1 Final Remarks 50

6 Conclusions and future work **51**

LIST OF FIGURES

2.1 A phylogenetic tree based on rRNA data. 4
2.2 Double helix DNA structure. 4
2.3 From DNA to Proteins. 5
2.4 Sequence split between exons (Ex) and introns (In). 7
2.5 The spectrum of a DNA sequence. 7

3.1 Logarithmic function 10
3.2 Entropy of an experiment with two possible outcomes. 11
3.3 Huffman-coding example. 12
3.4 LZ77 sliding window and output. 13
3.5 LZ78 dictionary and output. 13
3.6 Separation between modeling and coding in arithmetic coding. 13
3.7 Arithmetic encoding of a string. 14
3.8 Example of finite-context model: the probability of the next outcome, x_{t+1}, considering $M = 5$ last outcomes. 15

4.1 Simplified diagram of the proposed method. 22
4.2 Three-state model. 23
4.3 Three-state model related to inverted complements. 24
4.4 State synchronization error. 25
4.5 Entropies from six different states and final output. 26

5.1 Graphical representation of the synthetic sequence A. 32
5.2 Graphical representation of the synthetic sequence B. 32
5.3 Identification of the reference gene *BIG* in the synthetic sequence A. 33
5.4 Identification of the reference gene *BIG* in the synthetic sequence B. 34
5.5 Search performed with BLAST in synthetic sequence B, using *BIG* gene as a query. 35

5.6 Identification of the reference gene *AT4G16162* in the chromosome 4 of *Arabidopsis thaliana*. 36
5.7 Identification of the reference gene *jmj2* in the chromosome 1 of *Schizosaccharomyces pombe*. 36
5.8 Identification of the reference gene *Slc4a2* in the chromosome 5 of *Mus musculus*. 37
5.9 Identification of the reference gene *Slc4a2* in the chromosome 5 of *Mus musculus* using different parameters. 37
5.10 Identification of homolog reference gene *MYO3A* in different target organisms. 39
5.11 Identification of homolog reference gene *SPATA45* in different target organisms. 40
5.12 Identification of homolog reference gene *SYT1* in chromosome 15 of *Canis lupus familiaris*. 41
5.13 Identification of homolog reference gene *SYT1* in chromosome 12 of *Gorilla gorilla*. 42
5.14 Identification of homolog reference gene *SYT1* in chromosome 12 of *Pongo abelii*. 42
5.15 Identification of homolog reference gene *SYT1* in *Pan troglodytes* 43
5.16 Identification of reference region 4.1 in chromosome 4 of *Pongo abelii*. 48
5.17 Identification of reference regions 12.1, 12.2 and 12.3 in target chromosome 12 of *Pan troglodytes*. 49
5.18 Identification of reference region 7.2 in chromosome 7 of *Gorilla gorilla*. . . . 49

LIST OF TABLES

2.1 The genetic code table. 6

3.1 Example of resulting codes using Huffman coding. 12
3.2 Order-5 finite-context model example. 16
3.3 Order-5 finite-context model update example. 17

4.1 State 0 of order 5 FCM. 23
4.2 State 0 of order 5 FCM for inverted complements. 24
4.3 Model of a reference sequence. 25

5.1 Gene subsequences characteristics of the synthetic sequence A. 32
5.2 Gene subsequences characteristics of the synthetic sequence B. 33
5.3 Detected regions position using a threshold of 0.6 in synthetic sequence A. . . 34
5.4 Detected regions position using a threshold of 1.4 in synthetic sequence B. . . 34
5.5 Genes used as a reference and their respective organism/location. 35
5.6 Location of the identified regions using `NET-ASAR` and the parameters used in the process. 37
5.7 Location and length of the identified regions in chromosome 5 of *Mus musculus* with reference gene *Slc4a2*, using inappropriate parameters. 38
5.8 Reference genes used and corresponding target chromosome(s). 38
5.9 Location of identified regions by `NET-ASAR`, with reference gene *MYO3A*. . . . 39
5.10 Location of identified regions by `NET-ASAR`, with reference gene *SPATA45*. . . 40
5.11 Location of identified regions by`NET-ASAR`, with reference gene *SYT1*. 41
5.12 Identified regions and their length with reference gene *SYT1* in chromosome 15 of *Canis lupus familiaris*. 41
5.13 Identified regions and their length with reference gene *SYT1* in chromosome 12 of *Gorilla gorilla*. 42
5.14 Identified regions and their length with reference gene *SYT1* in chromosome 12 of *Pongo abelii*. 42

5.15 Modern human specific regions, compared to a *Neanderthal* and a *Denisovan* genomes, used in this work. 46
5.16 Condensed results of the search using `NET-ASAR` with the modern human specific regions as a reference and different organisms as target. 47

GLOSSARY

DNA Deoxyribonucleic Acid

A Adenine

C Cytosine

G Guanine

T Thymine

RNA Ribonucleic Acid

U Uracil

FCM Finite-Context Model

bpb Bits per base

CHAPTER 1

INTRODUCTION

Deoxyribonucleic acid (DNA) is the genetic material that contains the instructions needed for the function and development of an organism.

In recent years, there has been a great increase in the amount of sequenced DNA, made possible by the investments made in technology improvement and by the important role that DNA study plays in understanding the past, present and future of organisms.

With the increase in the number of sequenced genomes, a problem has surfaced: the great amount of data that needs to be stored. Data compression plays a crucial part in solving this problem. By having ways of discarding redundant information it is possible to reduce the size needed for storage, thus several methods for DNA compression have been proposed throughout the years (see, for example, [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]).

The ability to locate and identify similar sequences of DNA is extremely important in understanding, for example, the connections between different organisms and their common traits, such as common ancestors. Different tools have been implemented with this goal in mind, using different techniques and focusing on different DNA characteristics (see, for example, [11, 12, 13, 14, 15, 16, 17]).

Given the inherent connection between data compression and the information content of a string, could it be used to compare the information content of different sequences and locate similarities? That is the main question this thesis aims to answer.

The work presented in this thesis is the result of studying a method based on data compression to locate and identify DNA sequences with some degree of similarity. Broadly speaking, when a compression method is performed, one gets a metric regarding the compression rate obtained. In this case, this metric is used as a measure of similarity between sequences, enabling the ability to locate similar regions.

1.1 Main contributions

The main goal of this work was to study, develop and evaluate the use of a compression technique for DNA sequence search. As such, this work's main contributions are:

- Implementation of an alignment-free [18] technique to localize DNA subsequences with some degree of similarity, relatively to a large target sequence, using Finite-Context Models.
- Implementation of a visualization software for result analysis.
- Presentation of detailed experimental results of a scientific application of the method in:
  - Synthetic data.
  - Real data.

1.2 Thesis organization

This document is structurally divided into the following chapters:

- Chapter 2 provides introductory biological concepts as a basis for the presented work.
- Chapter 3 begins by presenting the concept of information theory as a basis for data compression. It also presents an overview of data compression techniques, including some applications/techniques in DNA sequence compression and, the main tool utilized by the presented method, Finite-Context Models.
- Chapter 4 includes the method implementation and description.
- Chapter 5 contains the results of this work, firstly using synthetic sequences, comparing some of the obtained results with another known method, then using real sequences, either exact or homolog genes, and, finally, an application of the method to ancient DNA.
- Chapter 6 presents some conclusions about the developed work and possible ideas for future work.

CHAPTER 2

BIOLOGICAL BACKGROUND

2.1 DNA Overview

Deoxyribonucleic acid, commonly known as DNA, is the genetic material of every organism and contains all the necessary instructions for its function and development. It is mostly located in the cell nucleus (nuclear DNA) but it can also be found in the mitochondria (mitochondrial DNA or mtDNA) (in eukaryotes, such as humans and fungi; prokaryotes, such as the bacteria and archaea groups, have DNA at the cytoplasm). DNA is also found in some viruses. Figure 2.1 depicts a phylogenetic tree based on the proposal by Woese *et al.* [19], where the separation of Bacteria, Archaea and Eukaryotes is clear.

Apart from the three domains, there are also viruses. Viruses have been defined as small infectious agents, which only replicate inside the living cells of other organisms [20]. They can infect all types of life forms and, in some cases, integrate the hostage genomes [21, 22, 23]. Their classification is still a controversial subject, as many believe that the discovery of the giant DNA viruses, known as megavirus [24], should represent a fourth domain of life [25, 26]. Although they are associated with the death of other organisms, they are considered a major factor in the increase of the genetic diversity of organisms [27].

DNA consists of two polynucleotide strands that spiral around each other forming a double helix [28]. The nucleotides that compose a strand are formed of a sugar (deoxyribose), a phosphate and one of four bases (adenine, cytosine, guanine and thymine, typically represented by A, C, G and T, respectively). These bases are joined together in pairs through an hydrogen bond, with the possible pairings being between a purine and a pyrimidine (adenine-thymine and cytosine-guanine). It is this base pairing that allows the two strands to connect. Figure 2.2 represents the double helix DNA structure.

Different combinations of bases and their order allows for the great diversity between species and even within the same species. The sequences of nucleotides can be separated into coding and non-coding regions, where the relevant information to protein production can be

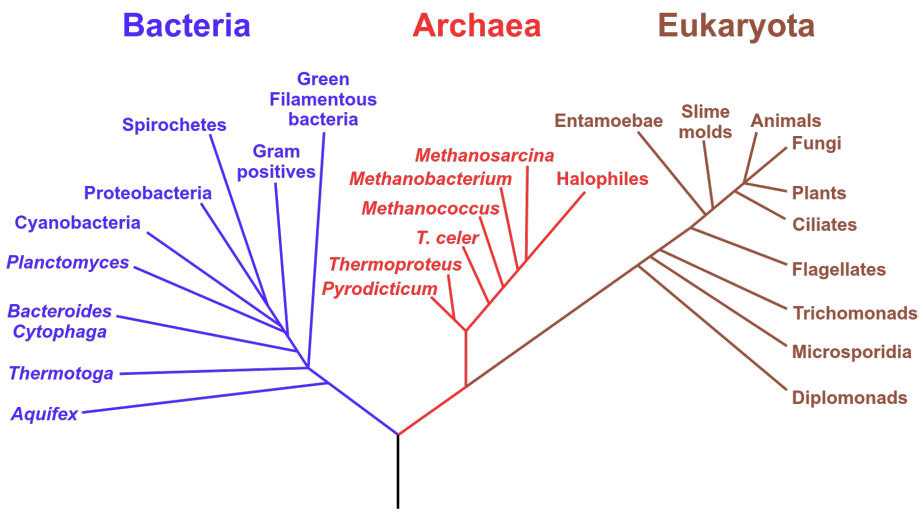


Figure 2.1: A phylogenetic tree based on rRNA data, showing the separation between the major branches (Bacteria, Archaea and Eukaryote). Source: wikipedia.org.

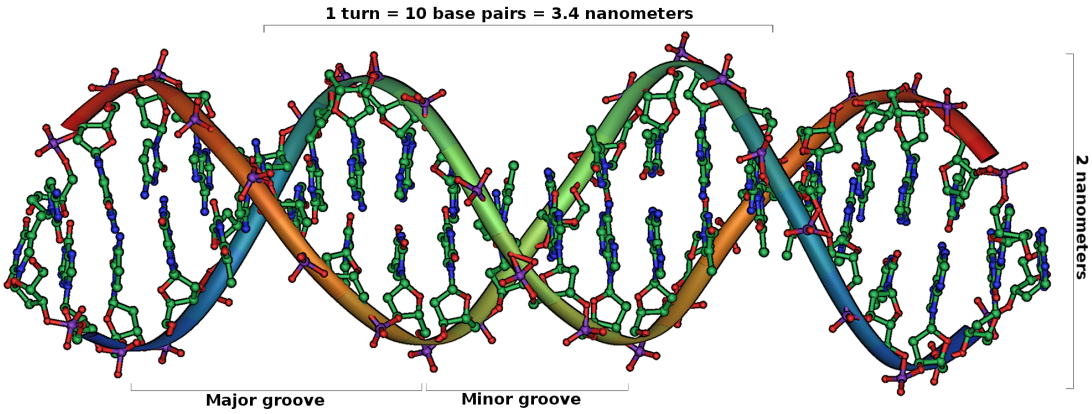


Figure 2.2: Double helix DNA structure. Source: wikipedia.org.

found in the coding regions. The process of unveiling DNA nucleotides is known as DNA sequencing [29].

The sequences that contain the information related to protein production are referred to as genes, with each gene consisting of parts that contain relevant information (exons) and parts that do not directly code proteins (introns) [30]. The concept of exons and introns only exists when referring to eukaryotes. Figure 2.3 represents the two main steps needed to produce proteins from the DNA coding regions: transcription and translation.

In the transcription step, RNA (Ribonucleic Acid) is made from DNA. RNA is synthesized in the nucleus and is similar to DNA, maintaining the use of bases, although, in RNA, no thymine (T) is used, with uracil (U) replacing it. Each sequence of RNA corresponds to the DNA sequence it was synthesized from.

Translation refers to the protein synthesis from the RNA sequence, following the genetic code, which is the set of rules by which the information contained in the mRNA (messenger

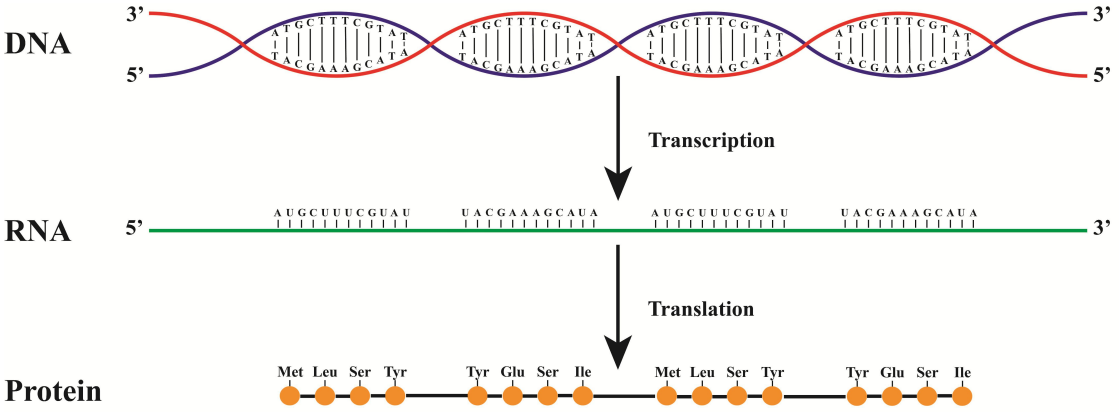


Figure 2.3: From DNA to Proteins [31].

RNA) is translated into proteins.

Proteins are formed by linking amino acids in the order specified in the genetic material, which is read in groups of three nucleotides, called codons [32]. Each codon specifies the next amino acid to be added to the protein strand.

Table 2.1 represents the genetic code. Of the possible 64 codons (4^3) only 61 direct amino acid incorporation into protein. The remaining three codons (UAA, UAG and UGA) are called the stop codons, which are involved in the termination of translation. The start codon (AUG) codes the amino acid methionine, and serves as a marker for the beginning of the translation.

Despite the number of codons (64), only 20 common amino acids are present in proteins, therefore, the same amino acid can be coded by up to six different codons, a property named code degeneracy.

As mentioned previously, the process of unveiling the genomic information into digital format is called sequencing. This process enables the possibility to analyze and study the genetic data. However, sequencing is not perfect, as, currently, it is only possible to segment fragments. As such, it is similar to finding the order and the content of a few pieces of a huge puzzle, that we want to assemble and analyze. The pieces of the puzzle, specifically tiny fragments of DNA (between 25 to 300 bases), might also contain several errors, and, as such, the analysis of the genomic sequences needs to be performed with tools/methods that are aware of such difficulties [33].

First Base	Second base: U		Second base: C		Second base: A		Second base: G		Third Base
U	UUU	Phenylalanine	UCU	Serine	UAU	Tyrosine	UGU	Cysteine	U
	UUC		UCC		UAC		UGC		C
	UUA	Leucine	UCA		UAA	Stop codon	UGA	Stop codon	A
	UUG		UCG		UAG	Stop codon	UGG	Tryptophan	G
C	CUU		CCU	Proline	CAU	Histidine	CGU	Arginine	U
	CUC		CCC		CAC		CGC		C
	CUA		CCA		CAA	Glutamine	CGA		A
	CUG		CCG		CAG		CGG		G
A	AUU	Isoleucine	ACU	Threonine	AAU	Asparagine	AGU	Serine	U
	AUC		ACC		AAC		AGC		C
	AUA		ACA		AAA	Lysine	AGA	Arginine	A
	AUG	Methionine (*)	ACG		AAG		AGG		G
G	GUU	Valine	GCU	Alanine	GAU	Aspartic acid	GGU	Glycine	U
	GUC		GCC		GAC		GGC		C
	GUA		GCA		GAA	Glutamic acid	GGA		A
	GUG		GCG		GAG		GGG		G

Table 2.1: The genetic code table. Each codon is translated to an amino acid, with the exception of the stop codons, but the same amino acid can be translated by up to six different codons. (*) The AUG codon is translated to methionine, and is also the start codon.

2.1.1 Inverted complements

DNA sequences frequently have subsequences that are the reversed complements of other subsequences that are also contained in the same sequence. Said subsequences are referred to as inverted complements. A complemented sequence is the sequence formed by the complemented bases of the original sequence, i.e. $A \leftrightarrow T$ and $C \leftrightarrow G$. A reversed sequence is, as the name states, the original sequence in reverse. Considering, as an example, the sequence "AGTAC", the inverted complement is then "GTACT".

In the scope of this thesis, inverted complements are an important occurrence, since genes are often found as inverted complements, so, while performing a search, it can be beneficial to apply this knowledge.

2.1.2 Three-base periodicity

As was previously mentioned, DNA contains coding and non-coding regions, with coding regions being surrounded by non-coding regions and even having exons separated by introns, effectively splitting up the sequence [34]. Figure 2.4 represents a sequence formed by exons and introns.

Three-base periodicity [35] is a property of coding regions and, as such, it can be useful to identify their location. This property can be seen as a preferential spacing between triplets by distances of three, six, nine, etc. bases [36].

The periodicity suggests the existence of three different, but related, information sources that drive a statistical model.

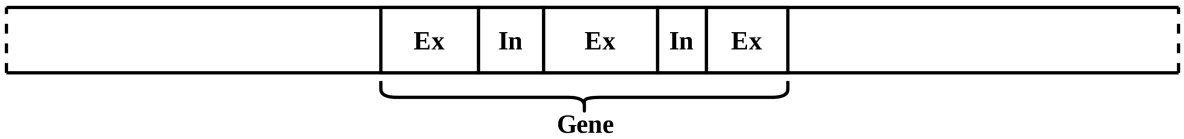


Figure 2.4: Sequence split between exons (Ex) and introns (In).

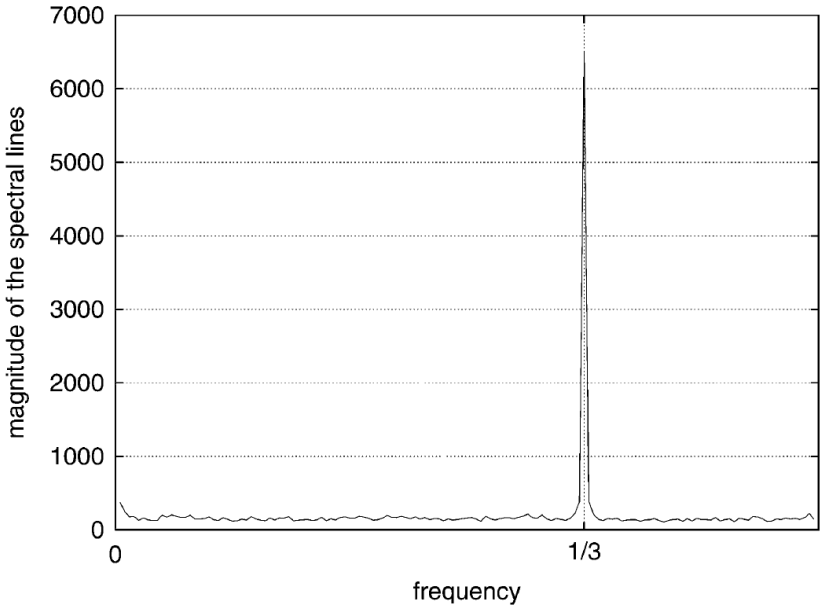


Figure 2.5: The spectrum of a DNA sequence (human chromosome 22), showing the period three ($f = 1/3$) [37].

Figure 2.5 illustrates the presence of the three-base periodicity in a DNA sequence (obtained from the human chromosome 22). The sharp peak at frequency 1/3 is associated with a period of length three.

CHAPTER 3

DATA COMPRESSION

3.1 Information Theory

Before introducing data compression, it is interesting to start by understanding the concept of information and relevant concepts about it. In 1948, Claude Shannon published the article "A Mathematical Theory of Communication", that stands as the founding work in the field of Information Theory. Shannon attempted to develop ways of quantifying the amount of information of a symbol or event without considering its meaning. He discovered the relation between the logarithmic function and information, and showed that an event with probability p has an amount of information equal to $-\log_b p$, denominated the self-information associated with that event.

By analyzing the logarithmic function as plotted in Figure 3.1, it is evident that an event with probability one leads to a value of information of zero; on the contrary, if the event has a probability that approaches zero, it can be said that it holds an amount of information approaching infinity. If $b = 2$, the information is measured in bits.

For data compression, there are two important principles to understand from this theory: entropy and redundancy.

A simple definition of entropy states that it is a real non-negative number proportional to the average minimum yes/no questions needed to get an answer to some given set of questions. It can also be seen as a quantification of the average amount of information present in a certain experiment.

Assuming an integer n between 1 and 32, how many yes/no questions are needed to find the number n? Considering, as an example, that $n = 27$, we need five questions:

1. $n \leq 16$? No;
2. $17 \leq n \leq 24$? No;
3. $25 \leq n \leq 28$? Yes;

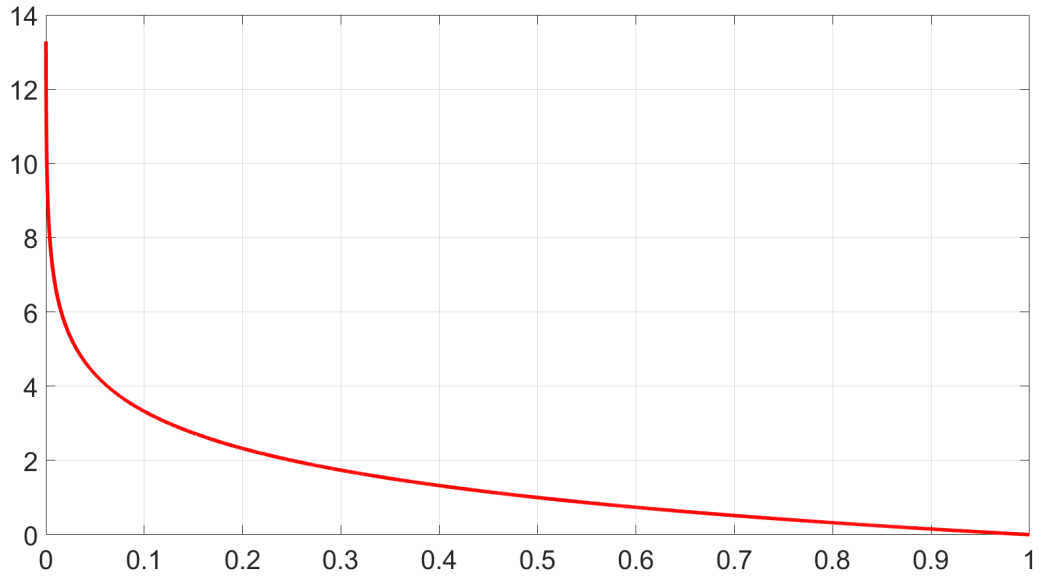


Figure 3.1: Logarithmic function $-\log_2(x)$, with $x \in [0, 1]$.

---

4. $25 \leq n \leq 26$? No;

5. $n = 27$? Yes;

The number of questions needed represents the number of times that 32 can be divided by 2, which is equivalent to $2^5 = 32$ or $\log_2 32 = 5$, showing the relation and importance of logarithms in information quantification.

Considering a random experiment X that can have N different values, each with a probability P_i, the average self-information, or entropy, of the random experiment is given by

$$H(X) = -\sum_{i=1}^{N} P_i \log_b P_i. \tag{3.1}$$

Consider now two symbols A and B, and their associated event probabilities P_A and P_B (with $P_A + P_B = 1$). Using Equation (3.1) and assuming equal probabilities ($P_A = P_B = 0.5$), an entropy of one is obtained, which means that a minimum of one bit is needed to encode each symbol.

Figure 3.2 shows the entropy of an experiment with two possible outcomes, in relation to the probability of either outcome. It can be seen that the maximum value of entropy in this experiment, which happens when the outcomes have equal probabilities, is also equal to one, showing the lack of redundancy, i.e., the data can not be compressed.

Assume now the case of a biased coin, where the probabilities of each face are $P_A = 0.7$ and $P_B = 0.3$. Calculating the entropy, a value of $H = 0.88$ is obtained: on average, each symbol can be encoded using 0.88 bits, which means that 100 coin tosses could be represented with approximately 88 bits, i.e., there is redundancy.

Redundancy is then defined as

$$R = \sum_i P_i \, l_i - \sum_i -P_i \log_2 P_i, \tag{3.2}$$

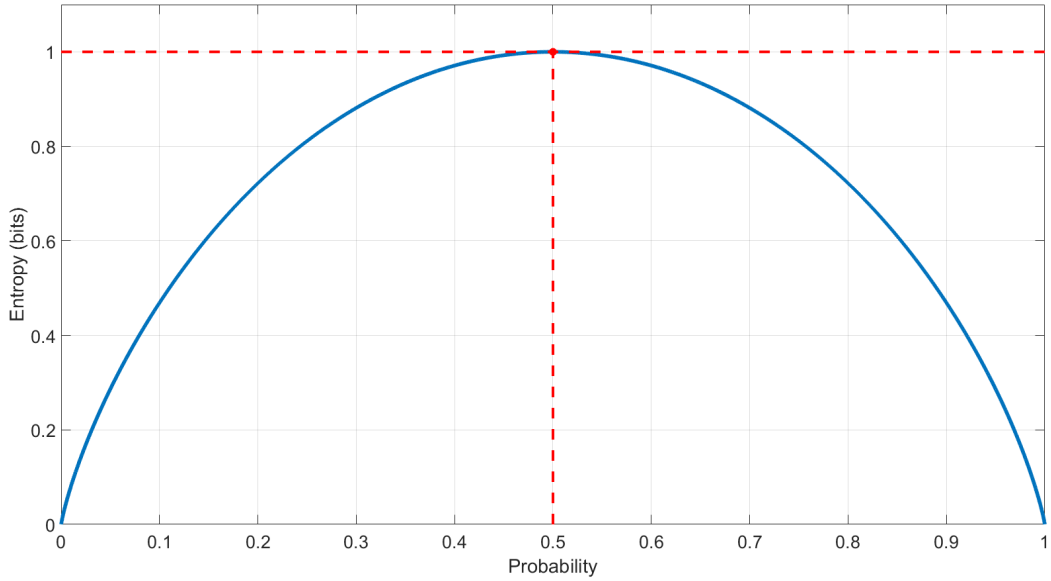


Figure 3.2: Entropy of an experiment with two possible outcomes.

where l_i represents the bit length of the code used to replace each symbol when compressing the data and $\sum_i P_i l_i$ represents the average code length (in the experiment above $l_i = 1$). From this expression, it can be concluded that the redundancy is zero when the average code length equals the entropy.

The redundancy can also be given by

$$R = \log_2(N) - \sum_i -P_i \log_2(P_i), \tag{3.3}$$

where N is the number of symbols.

3.2 Compression techniques

Compression is a technique that identifies and eliminates data redundancy, using a source coding method called compressor. Compression can be either lossless or lossy. With lossless compression, the compressed data can be decompressed to exactly their original values, since it does not lose information. On the other hand, lossy compression allows for some loss of information deemed nonessential, possibly making it impossible to obtain the exact original value after decompression.

There are several compression techniques, but most fall under three categories: variable-length coding, dictionary-based and arithmetic coding.

3.2.1 Variable-length coding

A variable-length code is a code that maps a source of symbols to a variable number of bits, such as the Shannon-Fano [38], Huffman [39] and Golomb [40] codes. The mapping of the source's symbols is done with different bit lengths for different symbols, according to their corresponding probabilities. The highest and lowest probability symbols are represented

using the smallest and biggest bit length codes, respectively, which, if done correctly, leads to compression.

Huffman coding is an optimum code [41] that serves as the basis for multiple applications, from compression formats like GZIP and PKZIP (based on the DEFLATE algorithm [42]), to image formats like JPEG [43]. Since Huffman codes are prefix-free (i.e., no code has, as prefix, the code from a different symbol), they allow immediate decoding.

Figure 3.3 represents an example of a Huffman code, for a source that generates four different symbols $\{a_1, a_2, a_3, a_4\}$ with different probabilities $P(a_1) = 0.4$, $P(a_2) = 0.35$, $P(a_3) = 0.2$, $P(a_4) = 0.05$. The codes corresponding to each symbol are shown in Table 3.1.

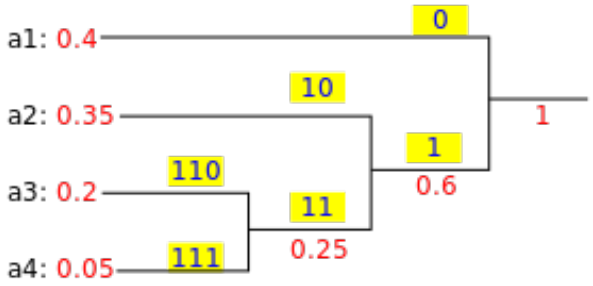


Figure 3.3: Huffman-coding example. Source: wikipedia.org.

Symbol	Code
a_1	0
a_2	01
a_3	110
a_4	111

Table 3.1: Example of resulting codes using Huffman coding.

3.2.2 Dictionary-based compression

A dictionary-based compressor (also known as a substitutional compressor) operates making use of a dictionary containing a set of strings, that can be static or dynamically built. As data are scanned, the occurrences are substituted by references to previous occurrences already present in the dictionary. Nowadays, most dictionary-based compression techniques are based on two methods, LZ77 [44] and LZ78 [45]. Although both methods use dictionary-based compression, they work differently.

LZ77 exploits the fact that words/phrases are likely to be repeated within a text file. It makes use of a sliding window, consisting of two portions (a search buffer and a look-ahead buffer), to examine the input sequence. The search buffer contains a portion of the recently encoded sequence while the look-ahead buffer contains the future portion to be encoded. The algorithm searches the sliding window for the longest possible match with the beginning of the look-ahead buffer, outputting, if possible, a pointer to the match. The output format can be, for example, a triplet formed by an offset to the match, the length of the match and

the next character to be encoded. An application of the LZ77 method can be found in gzip. Figure 3.4 represents an example of the sliding window used and the output produced.

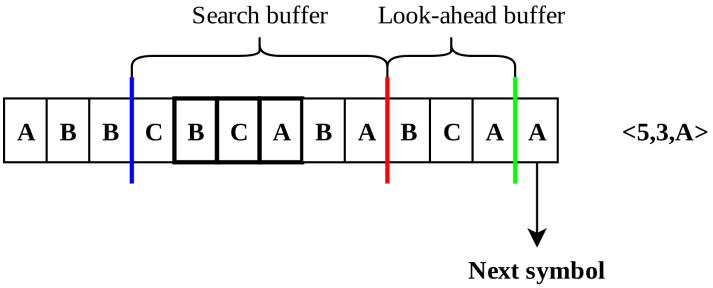


Figure 3.4: LZ77 sliding window and output.

The LZ78 method abandons the concept of a text window. It maintains an explicit dictionary and outputs in the form of pairs formed by an index referring to the longest matching dictionary entry and the first non-matching symbol. For each outputted pair, the sequence is also added to the dictionary, so that it can be referred to later. Nowadays, LZW [46], a variation of LZ78, is used, for example, in the GIF format. Figure 3.5 shows an example of a dictionary and corresponding outputs.

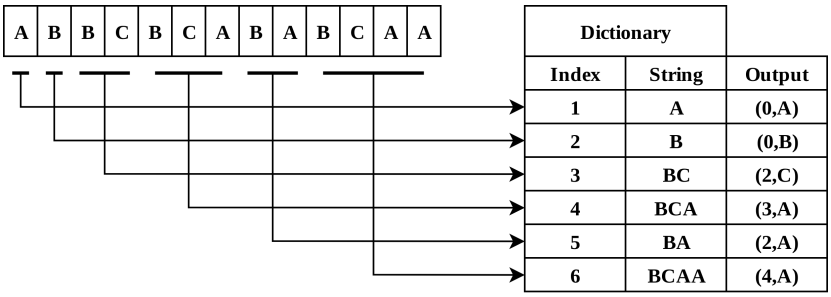


Figure 3.5: LZ78 dictionary and output.

3.2.3 Arithmetic coding

Arithmetic coding is the basis of the most recent compressors [47, 48]. When compression ratio is the main objective, arithmetic coding is arguably the most optimal entropy coding technique, since it usually leads to better results than Huffman Coding, while also allowing for a clear separation between modeling and coding [49]. The separation can be seen in Figure 3.6, while also showing the ability to use arithmetic coding with any statistical model.

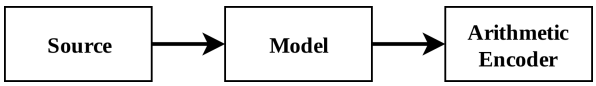


Figure 3.6: Separation between modeling and coding in arithmetic coding.

Arithmetic coding represents an entire message using a single code-word: a real number between zero and one. As the message becomes longer, the interval needed to represent the

message gets smaller and, therefore, a higher number of bits is needed to specify the interval. Successive symbols of the message reduce the interval size, with higher probability symbols leading to smaller size reductions and, consequentially, fewer bits being added to the message.

Consider an alphabet $\mathcal{A}$, composed of four symbols ($\mathcal{A} = \{A, B, C, D\}$), with fixed probabilities:

$$P(A) = 0.1, P(B) = 0.3, P(C) = 0.2 \text{ and } P(D) = 0.4.$$

Figure 3.7 shows a representation of the encoding process of the message "BCC".

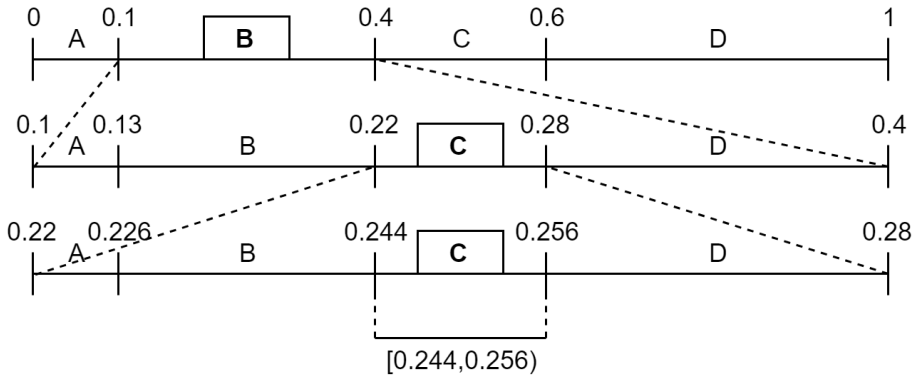


Figure 3.7: Arithmetic encoding of the string "BCC", assuming static probabilities.

Encoding

For each symbol, the interval is divided according to the respective symbol probability, e.g., when the first symbol, B, is seen, the encoder narrows the interval to [0.1,0.4), which represents the range allocated to this symbol, and divides this interval according to the probabilities of all the symbols. This process is repeated until the end of the message, resulting in the interval [0.244,0.256). The message can then be represented by any number contained in said interval; for example, 0.25.

Decoding

Suppose that the decoder knows that the message is represented by the number 0.25. Starting with the initial interval [0,1), the decoder can then deduce that the first symbol is B, since 0.25 lies within the interval allocated to B. Decoding a message follows a similar process as encoding, with the decoder finding the interval that contains the encoded message and the respective symbol after each interval size reduction.

3.3 Finite-context model (FCM)

Consider a source of information, s, that outputs symbols belonging to a finite alphabet, $\mathcal{A}$, and a sequence $x^t = x_1 x_2 x_3 \dots x_t$, that refers to the generated symbols at time t, as shown in Figure 3.8.

At any given time (or symbol), the finite-context model (FCM) assigns a probability estimate to the symbols of the alphabet $\mathcal{A}$. These estimates are calculated according to a conditioning context of finite size, M, of past symbols (order-M FCM) [50, 51], $c^t = x_{t-M+1}, \ldots, x_{t-1}, x_t$. The total number of possible contexts of an order-M finite-context model is, therefore, $|\mathcal{A}|^M$, which, in the scope of this thesis, is 4^M, since $|\mathcal{A}| = 4$ (referring to the four DNA bases).

In the case of genomic sequences, high orders usually lead to higher compression rates, although they require more time/computational resources.

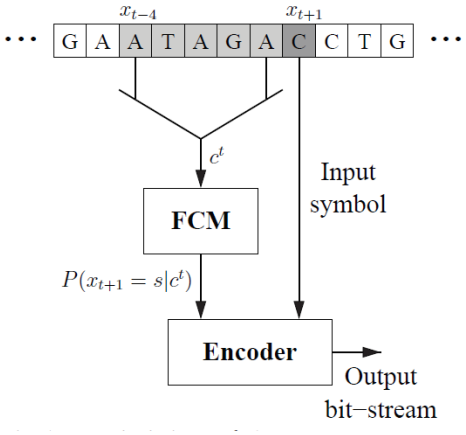


Figure 3.8: Example of finite-context model: the probability of the next outcome, x_{t+1}, considering $M = 5$ last outcomes [52].

The probability of x_{t+1} is then obtained by

$$P(x_{t+1} = s|\, c^t) = \frac{n_s^t + \alpha}{\sum\limits_{a \in A} n_a^t + 4\alpha}, \tag{3.4}$$

where n_s^t represents the number of past times that the symbol, s, has been generated after the conditioning context, c^t, and $\sum\limits_{a \in \mathcal{A}} n_a^t$ is the total number of times that the same context has appeared in the past. The parameter α is used to solve the problem of calculation of zero probability events, that is, when it is the first time that a certain symbol has appeared, which, with $\alpha = 0$ would lead to a zero probability, even though it is still possible for the symbol to occur. It is important to note that, generally, smaller values of alpha lead to better compression results when high order models are used. This is due to the fact that, when one decreases the value of alpha, the model is given a higher degree of certainty, since the impact of alpha in the probability of the symbol is diminished and, consequently, the impact of the conditioning context is increased. When combined with a higher order model, the resulting model is more confident (due to the low alpha value) and stores more information (due to the high order), which, in turn, leads to the aforementioned compression results.

FCMs are now easily translatable to tables that store the contexts, the number of times said contexts appear and the number of times a symbol appears given each context.

Table 3.2 presents a simple example of how an order-5 model is usually implemented. Each row presents the last five encoded symbols (context), the counter for each symbol and the sum of all the counters (total number of times the context has appeared in the sequence).

The counters are updated as each new symbol of the sequence is encoded. Considering that the last encoded symbols were "ATAGA" (c^t = "ATAGA") it is then possible for the model to obtain the following probability estimates:

$$P(A|\,ATAGA) = (1+\alpha)/(68+4\alpha),$$
$$P(C|\,ATAGA) = (22+\alpha)/(68+4\alpha),$$
$$P(G|\,ATAGA) = (31+\alpha)/(68+4\alpha) \text{ and}$$
$$P(T|\,ATAGA) = (14+\alpha)/(68+4\alpha).$$

Context, c^t	n_A^t	n_C^t	n_G^t	n_T^t	$\sum_{a\in\mathcal{A}}(n_a^t)$
AAAAA	4	12	2	23	41
AAAAC	18	5	11	17	51
⋮	⋮	⋮	⋮	⋮	⋮
ATAGA	1	22	31	14	68
⋮	⋮	⋮	⋮	⋮	⋮
GCAGT	9	19	21	29	78
⋮	⋮	⋮	⋮	⋮	⋮
TTTTT	8	3	6	15	32

Table 3.2: Order-5 finite-context model example.

It is known that the information content of the symbol x_{t+1} is given by $-\log_2 P(x_{t+1}|\,c^t)$ [51, 50] so, for a DNA sequence of N bases, the average bitrate (entropy) is given by

$$H_N = -\frac{1}{N}\sum_{t=0}^{N-1}\log_2 P(x_{t+1}=s|\,c^t)\,\text{bpb}, \tag{3.5}$$

where "bpb" stands for "bits per base".

Referring to the example presented in Table 3.2 and supposing that x_{t+1} (next symbol to encode) is "C", the model should now be updated with the new information. Table 3.3 represents the updated model.

It is important to note that the probability estimates of more probable events need less bits to be encoded than in the case of equal probabilities (where two bits are needed), e.g. $P(G|\,ATAGA) \approx 0.43$ bits (considering $\alpha = 1$).

Context, c^t	n_A^t	n_C^t	n_G^t	n_T^t	$\sum_{a\in\mathcal{A}}(n_a^t)$
AAAAA	4	12	2	23	41
AAAAC	18	5	11	17	51
⋮	⋮	⋮	⋮	⋮	⋮
ATAGA	1	**23**	31	14	**69**
⋮	⋮	⋮	⋮	⋮	⋮
GCAGT	9	19	21	29	78
⋮	⋮	⋮	⋮	⋮	⋮
TTTTT	8	3	6	15	32

Table 3.3: Order-5 finite-context model update example.

3.4 Genomic sequence compression

The quantity of DNA sequenced from organisms has increased rapidly and with it the need for compression, not only for storage purposes, but also for purposes of modeling and obtaining information on the sequences. Mutual compressibility has revealed to be important for discovering patterns of interest from genomes [53] and compression can be used as a good measurement of the relation between sequences [54, 55].

Genomic sequence compression presents itself as a process that requires specific purpose algorithms. This happens due to the characteristics of the sequences, such as the small alphabet (4 nucleotides), inverted complements (see Subsections 2.1.1 and 4.1.2) and their heterogeneous and non-stationary nature.

Genomic sequence compression falls mainly in two categories:

1. Individual compression;

2. Reference compression.

For detailed reviews on several methods and applications, see, for example, [56, 57].

3.4.1 Individual compression

Individual compression was firstly used as a way to decrease the space of transmitted data over the Internet. Currently, it is used because of space constraints and efficiency in data manipulations.

Commonly, there are a minimum of two different compression methods: one based on Lempel-Ziv substitutional procedures [44] and the other supported by low-order FCM arithmetic coding. As mentioned previously, substitutional procedures take advantage of repeated regions by representing them with a reference to past occurrences. It is also able to reference approximate repetitions, although, in this scenario, it is necessary to indicate the differences between the two regions.

For most methods, the low-order FCM assumes a secondary function, with the substitutional approach working as the main method. In the case of low performance by the substitutional method, the region is then represented by a low-order FCM.

The GeCo algorithm [58] uses a combination of context models of several orders for individual compression, as well as reference compression. This method introduces the concept of extended finite-context models (XFMCs), that allows for higher tolerance against substitution errors. For more flexibility, the method employs cache-hashes in high order models. The cache-hash uses a fixed hash function to simulate a specific structure, creating a middle point between a dictionary and a probabilistic model, but only takes in consideration the last hash entries in memory, in order to improve memory optimization.

The XM (eXpert Model) method [6], although also used for protein sequence compression, can be employed as an individual compression method. The method encodes each symbol by estimating the probability based on information obtained from the previous symbols. If the symbol is identified as part of a repeat, the information from one or more previous occurrences is used to determine the symbol's probability distribution. The symbol is then encoded using arithmetic encoding.

The DNAEnc3 method [5] introduces the use of several competing Markov models of different orders to obtain the probability distribution of the symbols, using them to obtain compression. This method allows the usage of models of orders of up to sixteen, a better handling of inverted repeats found in the sequences and provides probability estimates suited to the wide range of context depths used.

3.4.2 Reference compression

Reference compression was developed as a consequence of the big increase of sequenced genomes [59], with high redundancy characteristics, as well as the reduced costs of DNA sequencing.

The improvements of sequencing technology greatly increase the performance of reference compression [60]. With the DNA sequencing technologies improvements, it has become easier and more efficient to increase the quantity of sequenced DNA and, therefore, the number of available references. Consequently, better results can be achieved with reference compression.

The main idea of reference compression is taking advantage of similarities between the target sequence and a reference sequence (or several) and then encoding the differences between them. Since the decompressor also has access to the reference sequences it leads to very high compression rates [61].

GReEn (Genome Re-sequencing Encoding) [62] proposes the use of a probabilistic copy model. This method estimates the probability distribution of each symbol in the target sequence using either an adaptive model (the copy model), which assumes that the characters of the target sequence are an exact copy of reference sequence, or a static model that relies on the frequencies of the symbols in the target sequence. The adaptive model is the main statistical model but, in situations in which the adaptive model fails, the static model is used instead. The estimated probability distributions are then fed to an arithmetic encoder.

The iDoComp algorithm [63] is composed of three main steps. In the first step, called mapping generation, the goal is to create the parsing of the target sequence relative to the reference sequence, which can be seen as a sequence of substrings of the reference that, when concatenated, yield the target sequence. The second step, named post-processing of the mapping, tries to find consecutive matches that can be merged together and converted into an approximate match, leading to a reduction in the number of matches, and, consequently, the size of the mapping. Finally, the third step, named entropy encoder, further compresses the mapping and generates the compressed file.

The GRS method [8] is based on the diff utility from UNIX. The algorithm locates the longest subsequences that are similar in the reference and the target sequences, calculates a similarity measure and, if it is larger than an identified threshold, compresses the differences between the reference and target sequences using Huffman encoding. In the cases in which the similarity rate does not reach the threshold, the sequences are divided into smaller parts and the process is repeated using the new, smaller pieces.

The GDC 2 method [9], which is an improved version of GDC [64], uses two-level LZSS factoring [65] to encode the sequences as a list of matches and literals, having the advantage over GDC since it considers short matches after some longer ones.

3.5 Kolmogorov Complexity: the lower bound of compression

In 1965, Kolmogorov defined three different approaches for the problem of defining a complexity measure of a string: combinatorial, probabilistic and algorithmic [66]. The last on is known today as the Kolmogorov complexity. Specifically, the Kolmogorov complexity of x, $K(x)$, is given as the length of the shortest binary program that computes x on a universal computer (such as a universal Turing machine [67]) and then stops. However, this measure of Kolmogorov complexity is not computable, so it needs to be approximated using other computable measures [68, 69], namely a compressor.

One of the important problems that can be formulated using Kolmogorov theory is the definition of similarity, i.e., the information distance between objects. For example, on [70], it was proposed a similarity metric based on an information distance [71], defined as the length of the shortest binary program that is needed to transform strings A and B into each other. This distance depends on the Kolmogorov complexity of both strings, $K(A)$ and $K(B)$, and the conditional complexities, such as $K(A|B)$, i.e., the complexity of A when B is known.

According to [70], a compression algorithm needs to be *normal* [72] in order to be used to compute the normalized compression distance.

The concept of complexity measure provides an important applicability in the study of the functions of DNA, comparative analysis of organisms [73], genomic evolution and others [74, 75], through the use of DNA complexity profiles.

The main idea of this work is to approximate the Kolmogorov complexity of a string given a different string [76], filter and visualize the low complexity regions.

CHAPTER 4

COMPRESSION BASED SEQUENCE SEARCH

In this chapter, we propose a computational method (NET-ASAR), based on data compression, to search for a reference sequence in a target sequence.

The proposed method stands on finite-context models (FCMs). The main premise is that, by creating a statistical model of the reference sequence, and using said model to compute the entropy of each base (in bits) belonging to the target sequence, it should be possible to observe higher compression rates when the reference sequence, or a sequence with some degree of similarity, is present on the target.

The method stands on two main steps:

1. Modeling – A set of FCMs are loaded with the information of the reference sequence that, when concluded, are kept fixed until the end.

2. Entropy calculation – Using the previously loaded models of the reference, the entropy of each base of the target sequence is calculated using Equations (3.4) and (3.5).

Figure 4.1 represents the proposed method in a simplified manner and its main steps. The output is an ordered stream of bits per base (bpb), which can then be used to get a visual representation of the search result.

Two tools were developed in order to test the method. The first corresponds to the practical implementation of the method (NET-ASAR-map) and the second allows for easy visualization of results and, if requested, extraction of the regions that fall below a defined threshold (NET-ASAR-visual).

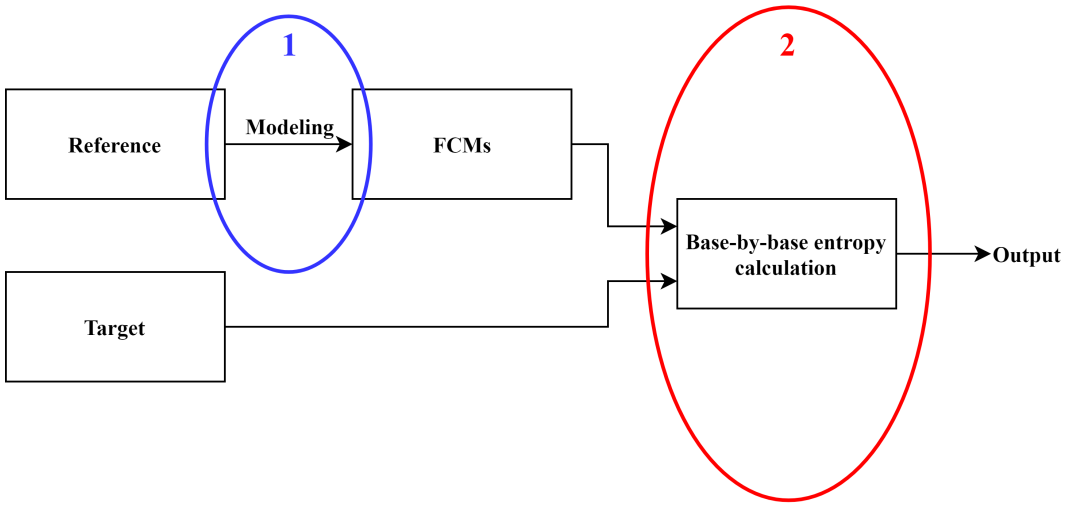


Figure 4.1: Simplified diagram of the proposed method.

4.1 The method

4.1.1 Three-state model

DNA coding regions have, as was mentioned earlier, an intrinsic property called three-base periodicity. In order to explore this property, the proposed method is implemented with a three-state model. This model differs from the model presented in Subsection 3.3 by the inclusion of three different states that are selected periodically and each state is comprised of a single finite-context model. Figure 4.2 shows a representation of the model.

By using this model, the probabilities not only depend on M but also on $(t \bmod 3)$, which is used for state selection. The probability estimator is then given by

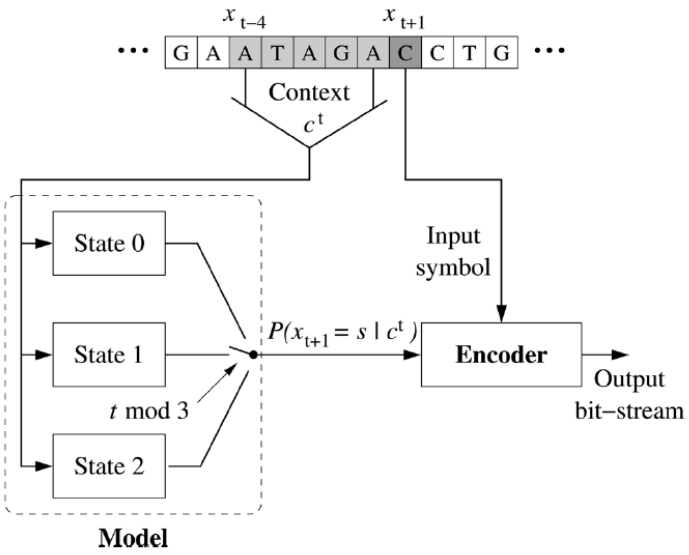
$$P(x_{t+1} = s|\, c^t) = \frac{{}_i n_s^t + \alpha}{\sum_{a \in \mathcal{A}} {}_i n_a^t + 4\alpha}, i = t \bmod 3. \tag{4.1}$$

It is easy to see the resemblance between Equation (4.1) and Equation (3.4), since a three-state model is inherently the combination of three different FCMs.

4.1.2 Inverted complements

When searching for a reference sequence, it is possible for it to be present in the target but as an inverted complement (see Subsection 2.1.1). In [52] the authors include the inverted complement of the context and complement of the symbol to encode when updating the counters of the model, so, in reality, the model will be updated twice for each symbol to encode. Taking into account the fact that a three-state model is being used, a new approach is needed, since the synchronization of states and bases cannot be assured in both cases (sequence/symbol and inverted complement sequence/complemented symbol).

The solution presented includes the usage of three extra states (corresponding to one extra FCM) that account for inverted complements. When encoding a new symbol, the model is updated as presented in Subsection 3.3 but, at the same time, the model referring to the



Note: In the scope of this thesis the block named "Encoder" is not used, as only the probability estimates are required.

Figure 4.2: Three-state model. The probability of the next outcome x_{t+1} is conditioned by the context depth, M, and $(t \bmod 3)$.

---

inverted complements is updated considering a conditioning context that equals the inverted complement of the original and the symbol to encode will be the complemented symbol located at x_{t-M} (immediately before the context).

The modeling can then be visualized as a sliding window of size M, where two different models are updated taking into account their three states. Considering the sequence presented in Figure 4.2 as the start of a reference sequence and ignoring the first two symbols ("GA"), we can assert that the modeling process is just starting (all model counters are set to zero). Figure 4.3 presents the corresponding sequence that will be considered when updating the model related to inverted complements.

Table 4.1 and Table 4.2 present the counters updated in both models when encoding the first symbol. Only the first state ($State\,0$) is presented, since it is the only one with counters different from 0.

Context, c^t	n_A^t	n_C^t	n_G^t	n_T^t	$\sum_{a\in\mathcal{A}}(n_a^t)$
AAAAA	0	0	0	0	0
⋮	⋮	⋮	⋮	⋮	⋮
ATAGA	0	**1**	0	0	**1**
⋮	⋮	⋮	⋮	⋮	⋮
TCTAT	0	0	0	0	0
⋮	⋮	⋮	⋮	⋮	⋮
TTTTT	0	0	0	0	0

Table 4.1: State 0 of order 5 FCM.

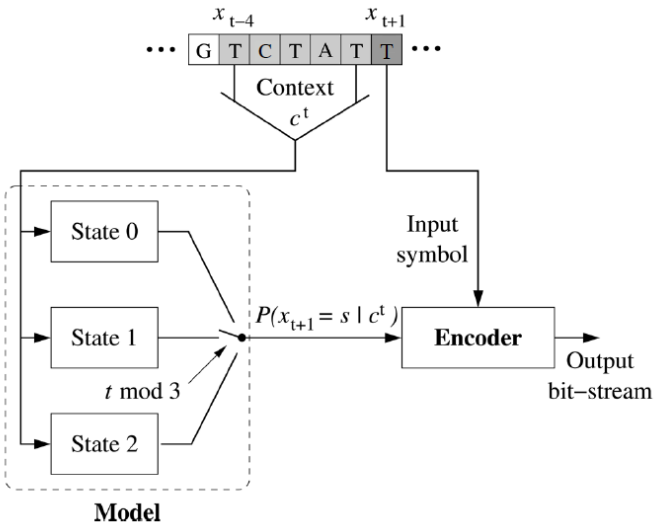


Figure 4.3: Three-state model related to inverted complements.

Context, c^t	n_A^t	n_C^t	n_G^t	n_T^t	$\sum_{a\in\mathcal{A}}(n_a^t)$
AAAAA	0	0	0	0	0
⋮	⋮	⋮	⋮	⋮	⋮
ATAGA	0	0	0	0	0
⋮	⋮	⋮	⋮	⋮	⋮
TCTAT	0	0	0	**1**	**1**
⋮	⋮	⋮	⋮	⋮	⋮
TTTTT	0	0	0	0	0

Table 4.2: State 0 of order 5 FCM for inverted complements.

4.1.3 Data models application

To proceed with the actual sequence search, it is necessary to calculate the entropy of each symbol, a, of the target sequence using Equation (3.5), knowing that the probability estimates are calculated using Equation (3.4), where each value of n_s^t and $\sum_{a\in\mathcal{A}} n_a^t$ is obtained by consulting the models created previously.

When searching in large sequences (e.g., chromosomes), it is important to remember that most information stored refers to non-coding regions, which may not carry three-base periodicity and do not necessarily have lengths multiples of three. Therefore, it can not be assumed that the states have the correct offset, i.e., that the three-base periodicity is not broken between exons. Since the modeling of the reference sequence always begins in state zero, a synchronization between the states is required, and, therefore, each base has to be modeled in the same state, in both reference and target.

Figure 4.4 shows an example in which the reference sequence is found in the target sequence, but preceded by an exon with two bases (represented as N). It can be clearly seen that, although the sequence is present in the target, the bases of the reference are represented in different states (highlighted in red), which would lead to wrong results.

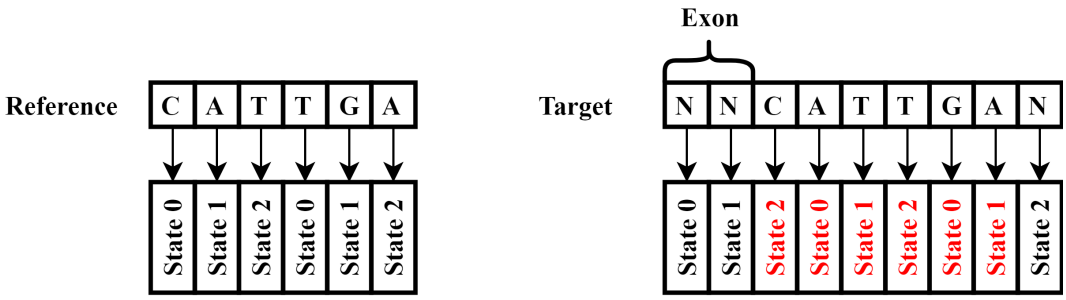


Figure 4.4: State synchronization error.

In order to overcome this issue, the calculations are performed three times for each model, in a way that the entropy of each symbol is obtained as the minimum of the entropy calculated using each different state, since, ideally, a lower entropy translates to the presence of the reference sequence, or a subsequence of the reference.

In reality, when inverted complements are included, the entropy of each symbol is given by

$$H_N = \min(H_{N_i}, H_{Ninv_i}), \text{ with } i = 0, 1, 2, \tag{4.2}$$

where H_{N_i} and H_{Ninv_i} represent the entropies of a base, calculated with the three states of "normal" and inverted models, respectively. It is evident that the entropy of each base refers to the minimum of six calculated entropies (when inverted complements are included in the search). Figure 4.5 presents hypothetic outputs from the six created FCMs and the resulting final output.

Consider now, as an example, the sequences "GATT" and "CTAGGATTCG", that are the reference and target sequences, respectively. For simplicity purposes consider, as well, that the reference sequence is being modeled using a single FCM of context size two ($M = 2$), and using $\alpha = 1/2$. Table 4.3 represents the model of the reference sequence.

Context, c^t	n_A^t	n_C^t	n_G^t	n_T^t	$\sum_{a\in\mathcal{A}}(n_a^t)$
AA	0	0	1	0	1
AG	1	0	0	0	1
GA	0	0	0	1	1
AT	0	0	0	1	1

Table 4.3: Model of a reference sequence.

Notice the context "AA", which is used as a way to include the first symbol in the modeling.

With the obtained model, it is now possible to perform the calculations needed to obtain

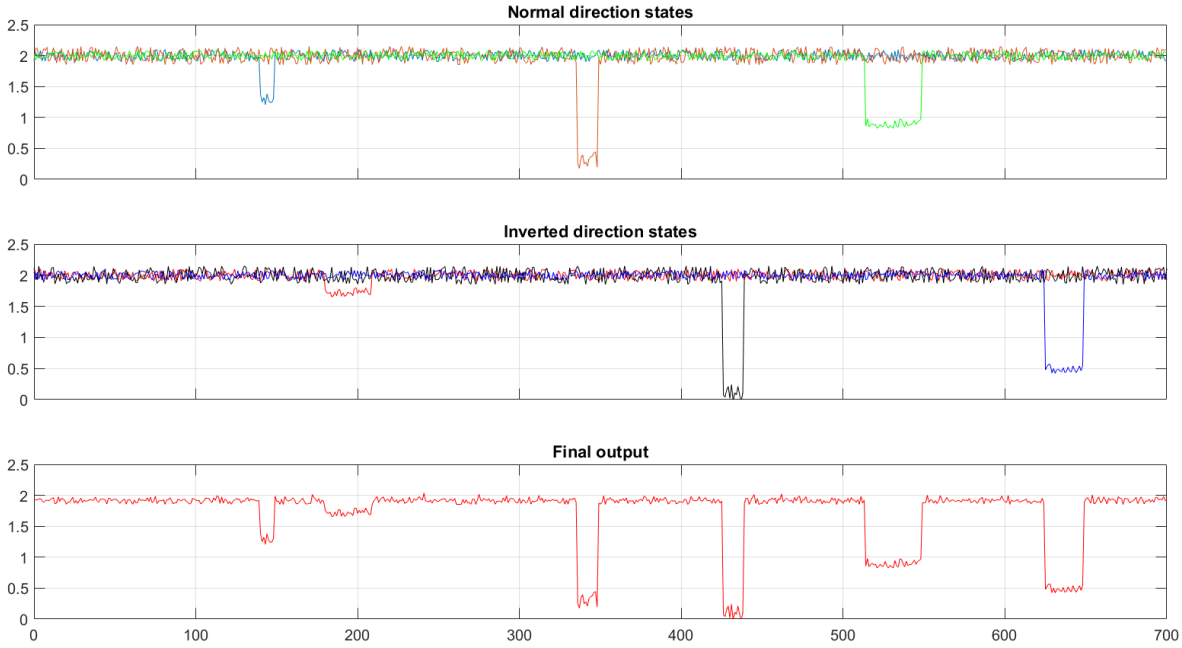


Figure 4.5: Entropies from six different states and final output, obtained as the minimum of the six (calculated at each base index).

---

the individual entropies of each base of the target sequence:

$$H(C|AA) = (0 + 0.5)/(1 + 4 \times 0.5) = 2.585\,\text{bits}$$
$$H(T|AC) = (0 + 0.5)/(0 + 4 \times 0.5) = 2\,\text{bits}$$
$$\dots$$
$$H(G|AG) = (0 + 0.5)/(1 + 4 \times 0.5) = 2.585\,\text{bits}$$
$$H(A|GG) = (0 + 0.5)/(0 + 4 \times 0.5) = 2\,\text{bits}$$
$$H(T|GA) = (1 + 0.5)/(1 + 4 \times 0.5) = 1\,\text{bit}$$
$$H(T|AT) = (1 + 0.5)/(1 + 4 \times 0.5) = 1\,\text{bit}$$
$$\dots$$

When looking at the calculated entropies, there is a clear drop in the bits needed to encode some of the bases of the target sequence, which indicates the presence of the reference sequence.

Although the needed bits are lower, this can only be said for two bases, with the first two bases of the reference sequence not being identified. This can be explained by the similarity between context size and reference sequence length, two and four, respectively, which, for the most part, should not be a problem when using the method with full DNA sequences (given the greater length of both the reference and target sequences).

4.1.4 Auxiliary tools

Given the nature of the obtained results, it becomes extremely useful to visualize them in a clear way, such as a graph of bits per base along the target sequence's bases.

The developed tool for result visualization (`NET-ASAR`-visual) takes the results previously filtered, a threshold, the segmented regions that fall below said threshold, and creates a graph. In this graph, the bits per base along the sequence are plotted, and a highlight of the segmented regions (regions below the threshold) is presented at the top of the plot.

The obtained subsequences are then extracted, if requested, from the original sequence and stored in files for further analysis.

The graph construction and subsequences extraction is achieved with the creation of different Bash scripts, a GNUplot script and tools available on the GOOSE toolkit[77].

Filtering

Although the filtering tool used during this work was implemented previously and is a part of the GOOSE toolkit, it is important to understand the parameters used.

The filtering is done using a sliding low-pass filter window, specifically, a Hamming window.

The parameters of the filter tool used in the presented work are the window size and the number of bases discarded. The number of discarded bases refers to the bases that are removed from the output, between each entry, i.e., if 20 bases are discarded it means that, when a filtered value is outputted, the next 20 bases are discarded and the next value is kept.

4.2 Implementation

`NET-ASAR` was implemented in the C++ language, using the FCMs with `unordered_maps`, found in the C++ Standard Library (`std`). A class was created in order to contain all variables and methods necessary for the algorithm.

For compiling, it was used the g++ compiler, present in Ubuntu (version 16.04) and the C++14 Standard.

The class that implements the FCMs can be reviewed in Listing 4.1. A structure named `values` is also included because, although it is not directly a part of the class, it is still necessary for the implementation of the class.

The method has been made publicly available, under GPLv3 license, at `https://github.com/manuelgaspar/NET-ASAR`.

```
struct values
{
        int val[5];
};

class modelClass
{
        private:
        const int context;
        std::string initialContext;
```

```
        public:
        const int alSize;
        const double alpha;
        modelClass(int cont, double alpha_);
        std::unordered_map<std::string,values> pMap[3];
        int getContext();
        std::string getInitialContext();

};
```

Listing 4.1: Finite-Context Model class header.

The implemented class has its own members: variables and methods. With the exception of context size and the initial context, all the word variables were set to public, since they need to be accessed regularly. The created variables are:

- `values` - a structure containing only an array of integers of size 5. This array will be used to store the counts, where `val[0]`, `val[1]`, `val[2]`, `val[3]` refer to 'A','C','G','T', respectively, and `val[4]` stores the count regarding the context.
- `context` - an integer that stores the size of the context, M.
- `initialContext` - a string that stores the initial context. The importance of this variable is evident when thinking of the first base to be encoded, since it does not have any context, one needs to be created, which, in this case will be a string with the symbol "A" repeated M times.
- `alSize` - an integer in which the alphabet size is stored so it can be used in the calculations. When the class is initialized this variable is automatically initialized with the value 4.
- `alpha` - a double that stores the value of alpha used in Equation (4.1).
- `pMap` - an array of `std::unordered_maps`. This variable is the most important variable of the class, since it is in this array that the models are stored. Note that the array is of size 3, referring to the 3 states of the model. Each map stores combinations of strings (*keys*) and `values`.

In regards to methods, the class contains the following:

- `modelClass(cont, alpha)` - constructor that stores the context size on the variable `context`, the alpha value in variable `alpha`, initializes the alphabet size (`alSize`) and creates the string `initialContext` with the specifications mention earlier.
- `getContext()` - returns the context size. It can be used to assert that the same context size is used throughout the algorithm.
- `getInitialContext()` - returns the string `initialContext`. It is used later on when the models are being created and when the calculations are made.

After the model class implementation, the main algorithm was implemented. It follows the next general steps when performing a typical search:

1. Using an `std::ifstream` to iterate through the reference sequence, base by base, the three-state models are created. While looping through the sequence and updating the corresponding counters, the contexts are updated by removing its first character and adding the previous base, in order to be used in the next iteration. It is worth noting that if the symbol does not belong to the alphabet used (i.e. $n_s^t \notin A, C, G, T$), the counters remain with the same value.

2. Iterating through the target sequence, base by base, the probability estimates are now calculated for each base using equations 3.4 and 3.5. This loop is performed three times in the interest of obtaining the probability estimates for each symbol with the three states. The minimum values of entropy obtained from the states are stored in a `std::vector<double>`.

3. Finally, the calculated values are then outputted to `std::cout`.

CHAPTER 5

RESULTS

5.1 Synthetic Sequences

In order to properly test NET-ASAR, it is important to start in a way that allows full control of the test conditions. For this reason, two synthetic sequences, both composed of randomly generated subsequences, were created:

- Sequence A, which contains a gene[1] repeated throughout its length, with different substitution mutation rates and orientation;
- Sequence B, that contains small sequences extracted from the gene[1], with varying degrees and types of mutation.

The gene is placed in the sequences, so that the previously mentioned DNA characteristics are taken into account when performing the search.

The synthetic sequences are represented in Figures 5.1 and 5.2, where the main rectangle represents the sequence and in red are represented the positions of the added gene information.

Table 5.1 shows the position, mutation (substitution) percentage and orientation of the gene in sequence A.

[1] Auxin transport protein (*BIG*), found in *Arabidopsis thaliana*

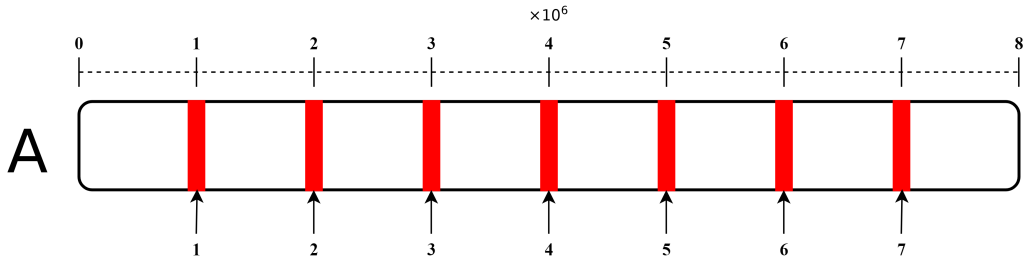


Figure 5.1: Graphical representation of the synthetic sequence A.

Sequence number	Location	Orientation	Substitution mutation (%)
1	1.000.001	Normal	0
2	2.017.805	Normal	1
3	3.035.609	Normal	2
4	4.053.413	Normal	3
5	5.071.217	Normal	4
6	6.089.021	Normal	5
7	7.106.825	Inverted Complement	0

Table 5.1: Gene subsequences characteristics of the synthetic sequence A.

Table 5.2 presents the characteristics of the used gene subsequences, such as the position it occupies in sequence B, the location in the gene and the degree of mutations (substitution and deletion).

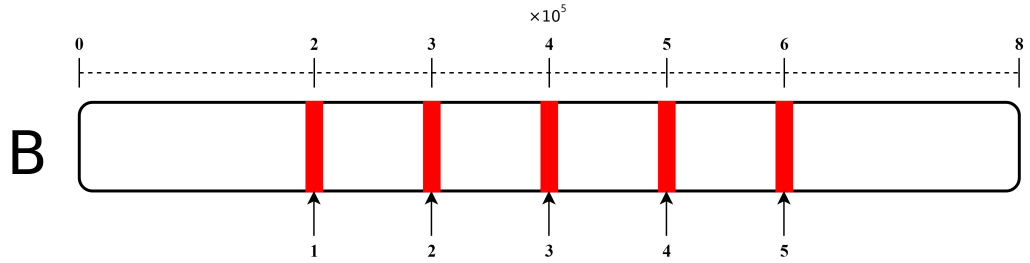


Figure 5.2: Graphical representation of the synthetic sequence B.

In order to validate the results, the search was also performed using BLAST (Basic Local Alignment Search Tool) [78], specifically, BLAST 2 Sequences [79], which is a BLAST-based tool for aligning two protein or nucleotide sequences. BLAST is an alignment algorithm that can detect sequence similarity between a query sequence and sequences within a database or library. In this particular case, the algorithm is used to find similarities between the gene and the synthetic sequence.

The comparison between methods was only performed using sequence B, since we felt that it would present a bigger challenge for both methods.

Sequence number	Location	Location in gene	Mutation (%)	
			Substitution	Deletion
1	200.001	0-100	2	3
2	300.098	100-232	3	2
3	400.229	400-500	5	5
4	500.326	11600-12000	5	10
5	600.680	15400-17000	5	15

Table 5.2: Gene subsequences characteristics of the synthetic sequence B.

5.1.1 Results

Figure 5.3 portrays the result obtained, using a context size of 12. Highlighted in red are the regions of the sequence that, when compressed with the reference sequence models, have an entropy lower than the threshold of 0.6 bits.

The choice of the threshold was made in a manner that allows for all the regions to be highlighted. This means that, if the chosen threshold would be, for example, 0.4 bits, the sixth region would not be highlighted, although, in this case, all the regions can easily be identified by the inverted peaks.

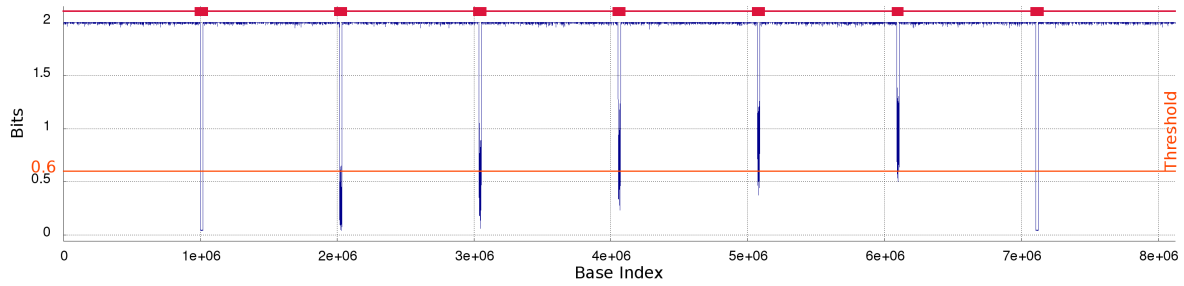


Figure 5.3: Identification of the reference gene *BIG* in the synthetic sequence A. Obtained with a context size of 12, a threshold of 0.6 and filtering with window size 200 and discarding 20 bases for each one kept.

Table 5.3 presents the positions highlighted by the method. In some cases, the segmented regions presented are the grouping of several small regions, found close together. The locations are represented this way, because, for some of the regions, the zones below the threshold are divided into several smaller subregions, i.e., in those regions we can find small variations in bits per base, which can lead to values above the threshold, followed by more values below the threshold.

Comparing Table 5.1 and Table 5.3, it is clear that the method has obtained the correct regions with a slight difference in the region position. The obtained difference can be explained by a combination of the mutations applied to the gene and filter used, since, when filtering, it is conceivable that lower entropy values are being filtered out, and, consequently, affecting the read positions.

Figure 5.4 shows the graph obtained when using sequence B as the target and the *BIG* gene as the reference sequence.

Sequence number	Location
1	1000062-1017744
2	2017890-2035404
3	3035886-3053337
4	4054470-4069317
5	5072403-5086851
6	6093423-6099954
7	7106904-7124565

Table 5.3: Detected regions position using a threshold of 0.6 in synthetic sequence A.

Using a threshold of 1.4 bits, leads to the highlighted regions in red, which correspond to the regions displayed in Table 5.4.

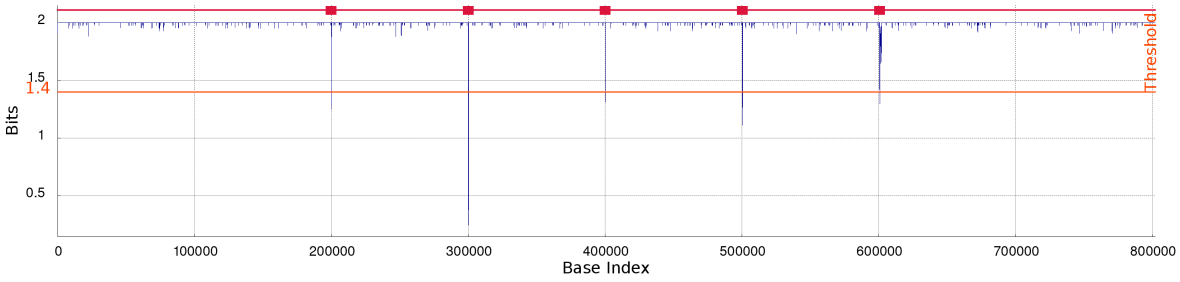


Figure 5.4: Identification of the reference gene *BIG* in the synthetic sequence B. Obtained with a context size of 12, a threshold of 1.4 and filtering with window size 75 and discarding 20 bases for each one kept.

Sequence number	Location
1	200025-200067
2	300111-300216
3	400281-400302
4	500367-500430
5	600726-600747

Table 5.4: Detected regions position using a threshold of 1.4 in synthetic sequence B.

Figure 5.4 exhibits, again, the ability, of the proposed method, to detect the regions, even when only small and mutated subsequences are present in the target sequence. Once again, the highlighted regions are not in exact agreement with the real positions. Note, that in some regions, by changing the used threshold, this difference can be reduced.

Consider now the results presented in Figure 5.5, which correspond to the previous search performed using BLAST [78].

A close analysis reveals that the BLAST algorithm was not able to detect all the regions, with subsequence number 1, located near the base of index 200.000 (see Figure 5.2), not

being detected, while the remaining four subsequences are correctly identified. Given the characteristics of the sequence, it is plausible that, by adjusting some of the parameters, the BLAST method could identify all the regions.

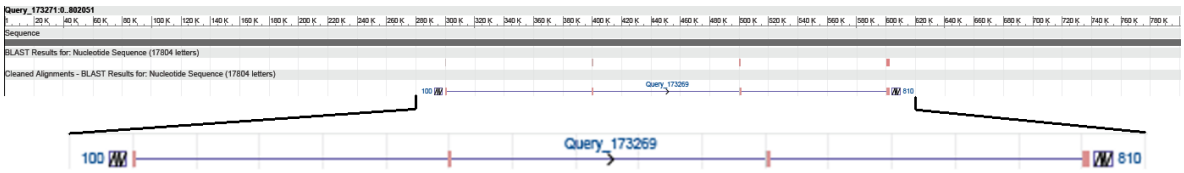


Figure 5.5: Search performed with BLAST in synthetic sequence B, using *BIG* gene as a query.

The results here presented demonstrate the proposed method's ability to detect and localize the presence of exact or approximate subsequences. The proposed method shows promising characteristics, when compared to a proven algorithm, as it was able to detect more regions than BLAST, although using more computational resources.

5.2 Real Sequences

Having evaluated the method using synthetic sequences, it should now be tested using less controlled condition, since, by using real sequences, it is expected that some regions of the reference sequence are repeated throughout the target, due to, for example, high repetition areas.

Firstly, results for exact gene search will be presented, by using, as target, the chromosome in which the gene (reference) is known to be present.

Secondly, the search is performed using target sequences that contain homolog genes.

5.2.1 Exact Gene Search

Table 5.5 shows the genes used for the exact gene search, as well as their location in the organism. The target sequence is, as previously mentioned, the chromosome containing each gene.

Gene	Organism	Chromosome	Location
AT4G16162	*Arabidopsis thaliana*	4	9159105-9161614
jmj2	*Schizosaccharomyces pombe*	1	1804548-1806797
Slc4a2	*Mus musculus*	5	24423761-24440947

Table 5.5: Genes used as a reference and their respective organism/location.

Figures 5.6, 5.7 and 5.8 show the obtained plots by `NET-ASAR`, with the highlighted regions. In Table 5.6 can be found a compacted view of the resulting regions and the parameters used for each search.

The obtained results are in accordance with the results obtained in Section 5.1.1, with each highlighted region differing slightly from the exact gene location.

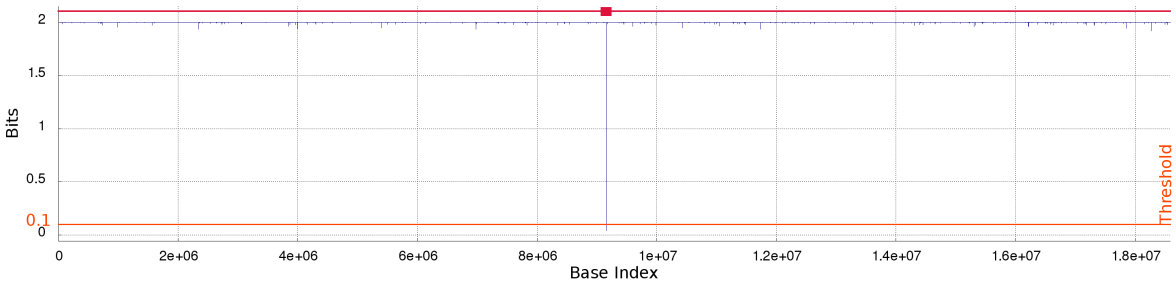


Figure 5.6: Identification of the reference gene *AT4G16162* in the chromosome 4 of *Arabidopsis thaliana*.

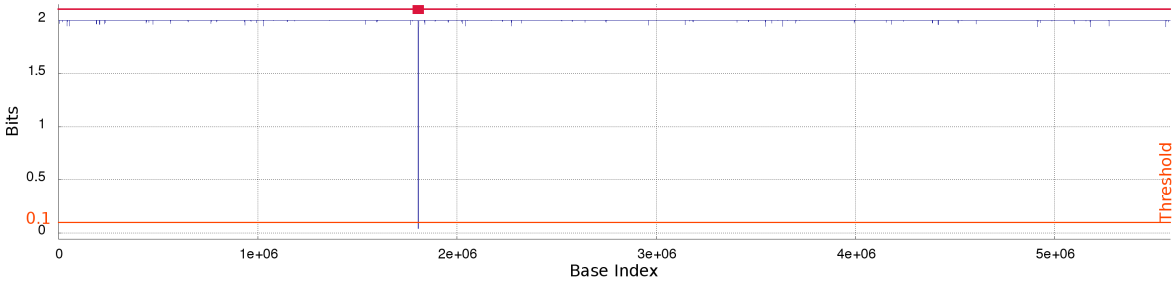


Figure 5.7: Identification of the reference gene *jmj2* in the chromosome 1 of *Schizosaccharomyces pombe*.

It is also interesting to observe the influence of the target sequence size. Chromosome 4 of *Arabidopsis thaliana*, chromosome 1 of *Schizosaccharomyces pombe* and chromosome 5 of *Mus musculus* have different sizes of approximately 18.6M, 5.6M and 151.8M (in number of bases), respectively. Looking at the presented plots, it is evident that, with the increase in size, the results are more prone to false positives, requiring further analysis of the regions.

Given the possibility of false positives, one can establish that the choice of context size, data filtering parameters, threshold and, to a certain extent, the α used in Equation (4.1), can impact the final results.

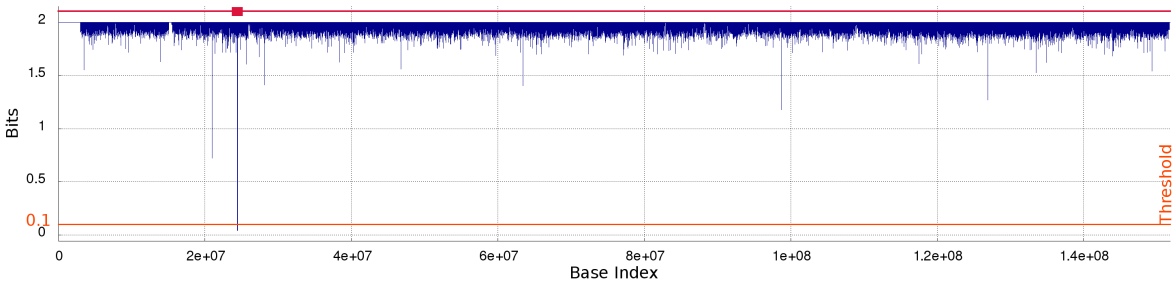


Figure 5.8: Identification of the reference gene *Slc4a2* in the chromosome 5 of *Mus musculus*.

Gene	Location	Context size	Threshold	Filtering	
				Window size	Discard
AT4G16162	9159507-9161229	18	0.1	500	20
jmj2	1804950-1806420	18	0.1	500	20
Slc4a2	24424155-24440556	18	0.1	500	20

Table 5.6: Location of the identified regions using `NET-ASAR` and the parameters used in the process.

Figure 5.9 shows the results obtained using different parameters and their effect. It was performed using a context size of 8, a filtering window of size 250, discard size of 20 and a threshold of 0.2.

Table 5.7 presents the highlighted regions referring to Figure 5.9 and the minimum value obtained in the region (inverted peak minimum). Of the three regions, only the second (highlighted) corresponds to the gene, with the remaining two (first and third) being false positives. Note the difference in length between the three, and how the highlighted region corresponding to the gene is the longest, by over four orders of magnitude.

When extracting the false positive regions and analyzing their contents it can be viewed that both are high pattern repetition regions (mostly "CA", in this particular case). Since the reference sequence contains an area with the same pattern repetition, albeit in a smaller number of repetitions, it translates to the obtained false positives.

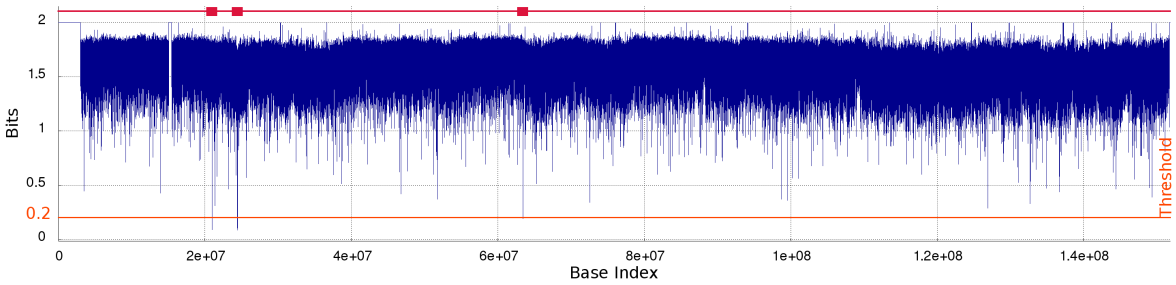


Figure 5.9: Identification of the reference gene *Slc4a2* in the chromosome 5 of *Mus musculus* using different parameters.

Subsequence	Location	Length	Minimum entropy
1	20978685-20979168	483	0.089
2	**24423924-24440766**	**16842**	**0.084**
3	63413070-63413217	147	0.192

Table 5.7: Location and length of the identified regions in chromosome 5 of *Mus musculus* with reference gene *Slc4a2*. The parameters used were a context size of 8, a threshold of 0.2, and filtering with window size 250 and discarding 20 bases. The highlighted subsequence refers to the region in which the gene can be found.

When the search is performed on exact matches between reference and target sequences, i.e., the reference sequence is a subsequence of the target sequence (as the previously presented examples), using high order models provides better outcomes, but when lower context sizes are used the results need to be analyzed with extra care.

5.2.2 Homolog genes

As was previously mentioned, the proposed method was tested using homolog genes.

Homology is, usually, associated with evolution, so, when referring to homolog genes, it is referring to similar genes that are present in different organisms that share a common ancestor.

Knowing this similarity between sequences, `NET-ASAR` can be tested using slightly different sequences by searching a reference gene in a target chromosome of a different organism, in which a homolog gene of the reference is present.

Table 5.8 presents the reference sequences (genes) and the corresponding target sequences (chromosomes). Three different genes were used as a reference, all extracted from the *Homo sapiens* genome, with each gene being used as a reference for different target chromosomes. When available in the NCBI databases [80], it is also presented the DNA Identity percentage, between the reference gene and the homolog gene present in the target sequence.

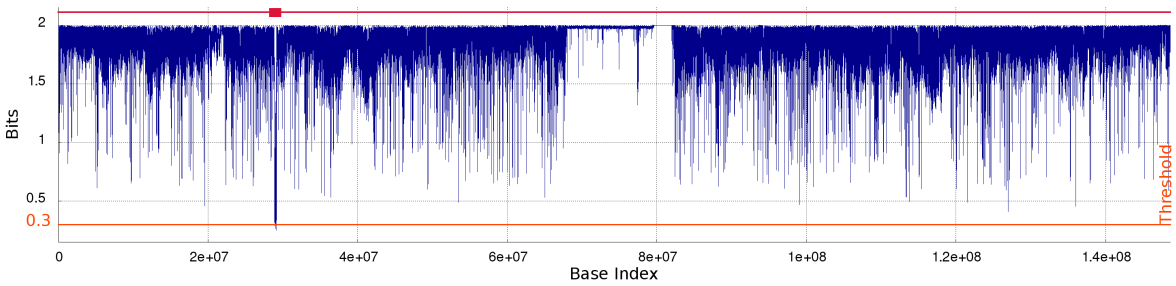
Reference Sequence		Target Sequence		DNA Identity (%)
Organism	Gene	Organism	Chromosome	
Homo Sapiens	*MYO3A*	*Gorilla gorilla*	10	–
		Pan troglodytes	10	99.3
	SPATA45	*Macaca mulatta*	1	96.6
		Pan troglodytes	1	99.0
	SYT1	*Canis lupus familiaris*	15	92.7
		Gorilla gorilla	12	–
		Pongo abelii	12	–
		Pan troglodytes	12	99.7

Table 5.8: Reference genes used and corresponding target chromosome(s).

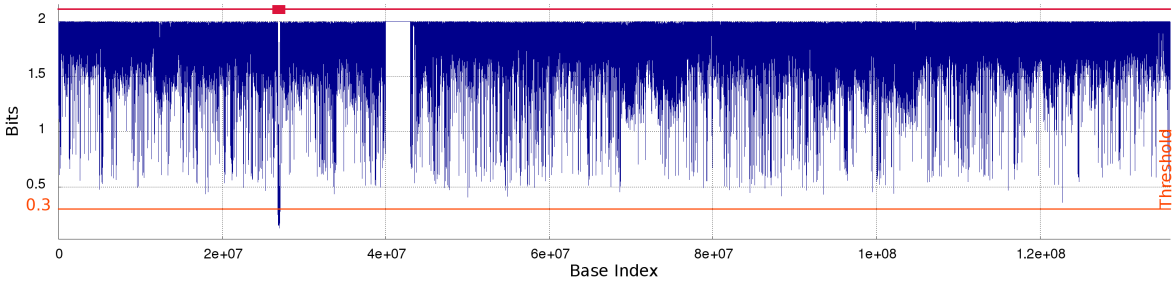
Results for gene MYO3A

Target organism	Results		Parameters used		
	Location	Figure	Filtering	Context	IR[2]
Gorilla gorilla	28940121-29114736	5.10a	-d 20 -w 2000	13	Yes
Pan troglodytes	26822607-27110853	5.10b	-d 20 -w 1000	13	Yes

Table 5.9: Location of identified regions by `NET-ASAR`, with reference gene *MYO3A* in different organisms and parameters used for the identification.



(a) Chromosome 10 of *Gorilla gorilla*.



(b) Chromosome 10 of *Pan troglodytes*.

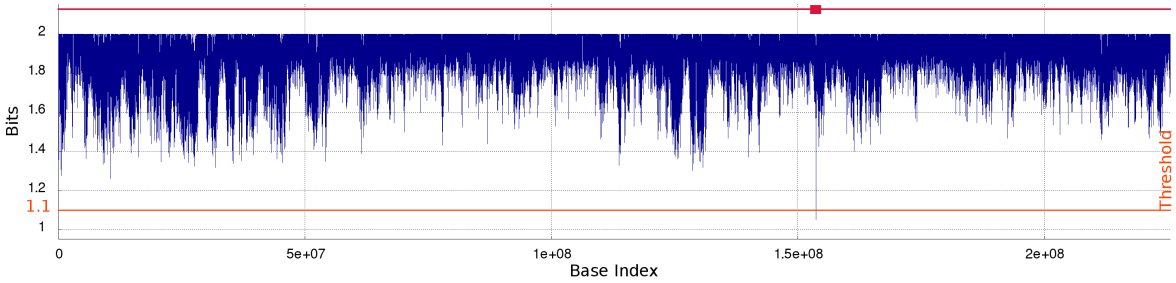
Figure 5.10: Identification of homolog reference gene *MYO3A* in different target organisms.

[2]Include inverted repeats in the search

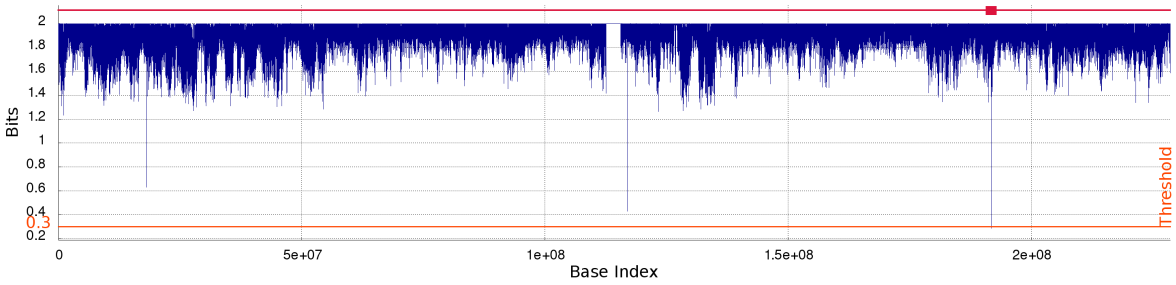
Results for gene SPATA45

Target organism	Results		Parameters used		
	Location	Figure	Filtering	Context	IR
Macaca mulatta	153695703-153699378	5.11a	-d 20 -w 2000	13	Yes
Pan troglodytes	191790081-191792874	5.11b	-d 20 -w 2000	13	Yes

Table 5.10: Location of identified regions by `NET-ASAR`, with reference gene *SPATA45* in different organisms and parameters used for the identification. The parameters -d and -w refer to the number of discarded bases for each base kept and the filtering window size, respectively.



(a) Chromosome 1 of *Macaca mulatta*.



(b) Chromosome 1 of *Pan troglodytes*.

Figure 5.11: Identification of homolog reference gene *SPATA45* in different target organisms.

Results for gene SYT1

Target organism	Results		Parameters used		
	Location(s)	Figure	Filtering	Context	IR
Canis lupus familiaris	Table 5.12	5.12	-d 20 -w 500	13	No
Gorilla gorilla	Table 5.13	5.13	-d 20 -w 1000	13	Yes
Pongo abelii	Table 5.14	5.14	-d 20 -w 1000	13	No
Pan troglodytes	81263994-81862452	5.15	-d 20 -w 1000	13	No

Table 5.11: Location of identified regions by`NET-ASAR`, with reference gene *SYT1* in different organisms and parameters used for the identification. The parameters -d and -w refer to the number of discarded bases for each base kept and the filtering window size, respectively.

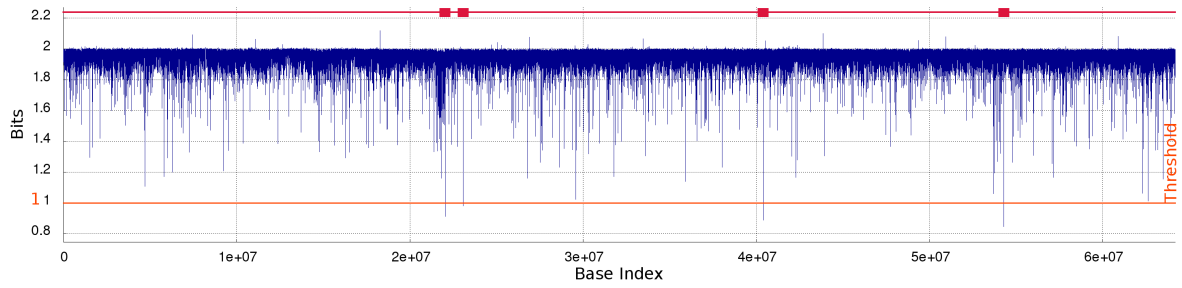


Figure 5.12: Identification of homolog reference gene *SYT1* in chromosome 15 of *Canis lupus familiaris*.

Subsequence	Location	Length	Minimum entropy
1	**22039962-22040235**	**273**	**0.911**
2	23082843-23082927	84	0.98
3	40416096-40416306	210	0.888
4	54298188-54298482	294	0.845

Table 5.12: Identified regions and their length with reference gene *SYT1* in chromosome 15 of *Canis lupus familiaris*. The highlighted subsequence refers to the location of the homolog gene of *SYT1*.

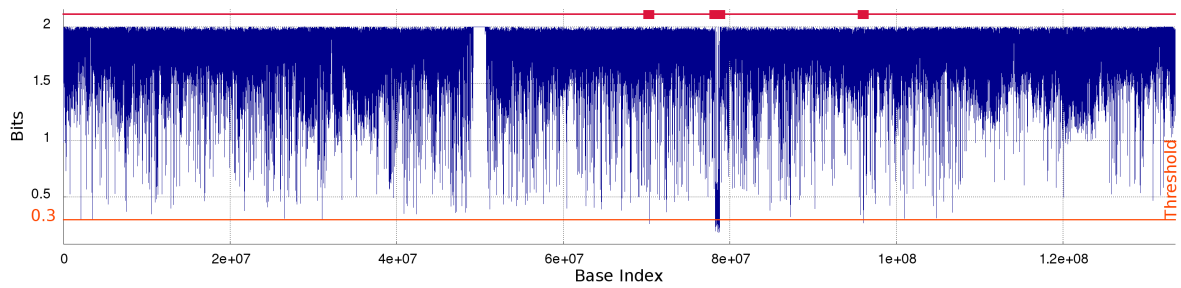


Figure 5.13: Identification of homolog reference gene *SYT1* in chromosome 12 of *Gorilla gorilla*.

Subsequence	Location	Length	Minimum entropy
1	70287861-70289352	1491	0.264
2	**78213051-78792252**	**579201**	**0.185**
3	96019434-96019938	504	0.273

Table 5.13: Identified regions and their length with reference gene *SYT1* in chromosome 12 of *Gorilla gorilla*. The highlighted subsequence refers to the location of the homolog gene of *SYT1*.

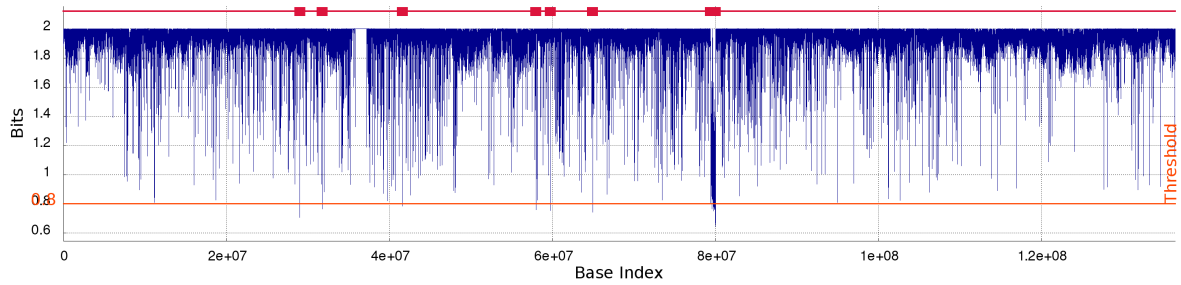


Figure 5.14: Identification of homolog reference gene *SYT1* in chromosome 12 of *Pongo abelii*.

.

Subsequence	Location	Length	Minimum entropy
1	29004864-29006439	1575	0.705
2	31780287-31781022	735	0.765
3	41625675:41626095	420	0.785
4	57980328:57981231	903	0.758
5	59733219:59734290	1071	0.753
6	64945776:64946868	1092	0.743
7	**79383885-79978458**	**594573**	**0.644**

Table 5.14: Identified regions and their length with reference gene *SYT1* in chromosome 12 of *Pongo abelii*. The highlighted subsequence refers to the location of the homolog gene of *SYT1*.

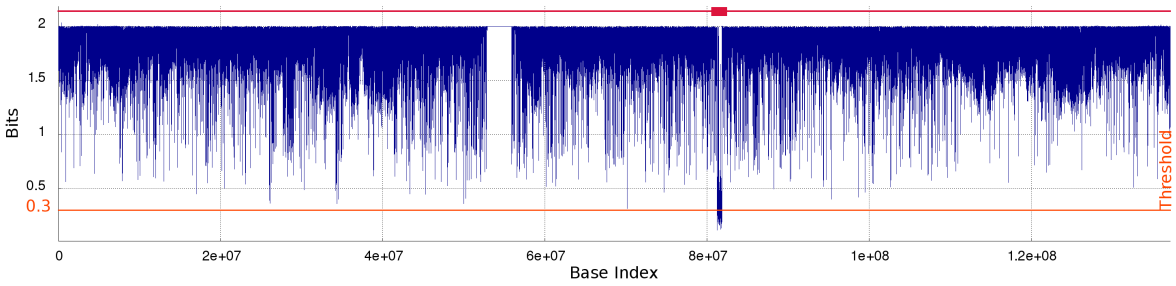


Figure 5.15: Identification of homolog reference gene *SYT1* in chromosome 12 of *Pan troglodytes*.

Analyzing the figures presented, it can be said that `NET-ASAR` is able to identify and localize similar sequences, such as homolog genes.

Although the method locates the reference sequences, it is also evident that it is necessary to analyze the graphs and located regions. The extra analysis is extremely important, because several factors can affect the ability to correctly differentiate between positives and false positives, such as the degree of similarity between the reference and the target, the length of the reference sequence and the parameters used.

When using `NET-ASAR` with highly similar sequences, the identified regions are easily recognizable, as seen in, for example, Figures 5.10b and 5.15, which have an identity of 99.3% and 99.7%, respectively.

Taking into account the size of the reference sequence, it can be said that `NET-ASAR` provides clearer results when using smaller references, as seen in Figure 5.11, as the reference gene *SPATA45* is, approximately, 17.5k bases long, in contrast with *MYO3A* (∼278.5k) and *SYT1* (∼588.0k). This occurrence can be viewed as a consequence of the decrease in size of the FCMs created, i.e., the number of entries of the models is smaller, and, potentially, even leading to fewer contexts represented in the model.

Considering the case of the reference gene *SYT1*, which is the biggest gene of the three tested, it is clear that the occurrence of false positives is more likely, as can be seen in Figures 5.13 and 5.14. Since several regions are identified, a closer look is necessary. Tables 5.13 and 5.14 present the regions detected in *Gorilla gorilla* and *Pongo abelii*, respectively. Comparing the regions, it is clear that, in both tables, one of the subsequences refers to a positive result, while the rest are false positives, since their length is extremely bigger.

Inspecting, now, Figure 5.12, it is visible the effect of a lower identity (92.7%), combined with the large reference gene. If one were to simply take the information from Table 5.12, such as the length of the sequences and the minimum value entropy they reach, it would be easy to make the assumption that the reference gene is present on subsequence 4, as it is the biggest while having a lower entropy minimum. However, when the subsequences are extracted, we are able to deduce what subsequence corresponds to the correct location of a similar sequence to the reference (subsequence 1, as highlighted in the table) and that the remaining three are false positives, caused by low complexity/high repetition regions.

Overall, the reference homolog genes were identified by the method, even though some sequences can not be identified automatically and require some extra analysis in order to differentiate positives and false positives.

5.3 Application to ancient DNA

Having assessed the method's capability of identifying reference sequences, with a certain similarity in relation to a region of a target sequence, using known conditions, with either synthetic or real sequences, the method can be used for scientific research purposes.

Using `CHESTER` [81, 82], several modern human specific regions were found. Modern human specific regions are DNA sequences that are unique or share high dissimilarity rates to close species, such as evolutionary regions.

The human specific regions detected were obtained when comparing the modern human genome to a *Neanderthal* [83] and a *Denisovan* high coverage [84] genomes, simultaneously, using a Bloom filter with a size of 64 GB ($m = 549,755,813,888$), k-mer size of 30 for the mapping (`CHESTER-map`), a window size of 101, threshold of 0.5 for the filtering and segmenting (`CHESTER-filter`). Therefore, the detected regions are regions that can be found in the modern human genome, but not on the *Neanderthal* and *Denisovan* genomes (or are found with high dissimilarity rates).

After having identified the modern human specific regions, `NET-ASAR` was used to detect and locate them in three modern target organisms: *Pan troglodytes*, *Gorilla gorilla* and *Pongo abelii*.

Since the number of detected regions was high, it would take too much time and resources to process every region. For this reason, only some of the regions were used, which we have selected based on a minimum length of 600 bases and presence of identified genes. This decision has left us with 31 regions, which can be found in Table 5.15. For each region, it can be seen the chromosome of the modern human genome in which the regions can be found, the designated name (used for referring each region, when needed), their location and length and the symbol and type of the gene found in the region.

The genes located in the regions are:

- DPPA2 upstream binding RNA (344595);
- zinc finger protein 385D (79750) - a candidate gene for reading disability and language impairment [85];
- fibroblast growth factor 12 (2257) - silencing of this gene showed significant inhibition in activity of tumor cell proliferation, colony formation, and cell migration. The gene has a potential role in esophageal squamous cell carcinoma [86];
- pyroglutamylated RFamide peptide receptor (84109) - possible new candidate involved in the etiology of late-onset Alzheimers disease [87];
- follistatin like 4 (23105) - has been associated with coronary heart disease. In white Cardiovascular Health Study participants was nominally associated with increased risk of stroke [88];
- uncharacterized LOC105378113 (105378113);
- contactin associated protein like 2 (26047) - variants have been associated with multiple neuropsychiatric conditions such as schizophrenia, autism spectrum disorder and intellectual disability. Animal studies indicate a role for CNTNAP2 in axon guidance, dendritic arborization and synaptogenesis. [89, 90];

- STEAP2 antisense RNA 1 (100874100) - the *STEAP2* gene's expression may be a crucial molecule in driving the invasive ability of prostate cancer cells [91];
- glutamate rich 1 (157697) - might be involved in the development/progression of Pancreatic ductal adenocarcinoma [92];
- uncharacterized LOC105379389 (105379389);
- MRGPRG antisense RNA 1 (283303);
- uncharacterized LOC105376678 (105376678);
- deleted in lymphocytic leukemia 1 (non-protein coding) (10301) - is involved in many solid tumours and haematological malignancies. May in part function as a tumor suppressor gene and confer chemoimmunotherapy resistance in children and adolescents with Burkitt lymphoma [93, 94];
- ATPase phospholipid transporting 10A (putative) (57194) - exogenous expression alters the lipid composition at the plasma membrane, which may in turn cause a delay in cell spreading and a change in cell morphology [95];
- thrombospondin type 1 domain containing 4 (79875) - may play key roles in tumorigenesis and development of esophageal carcinoma involved in cell cycle and Endocytosis [96];
- uncharacterized LOC107985161 (107985161);
- uncharacterized LOC105373032 (105373032);
- Kruppel like factor 8 (11279) - plays a key role in cancer progression and serves important roles in tumorigenesis and tumor progression [97, 98];
- testis expressed 11 (56159) - is mutated in infertile men with non-obstructive azoospermia. Mutations in this gene, including frameshift and splicing acceptor site mutations, cause infertility in 1% of azoospermic men [99];
- acyl-CoA synthetase long chain family member 4 (2182) - dysregulation is related to a great number of malignant tumors. This gene is overexpressed in hepatocellular carcinoma, and it will be a new potential therapeutic target for hepatocellular carcinoma as an independent adverse prognostic parameter [100].

It is worth noting that all searches were performed using a context size of 12, with a filtering window of 300 bases and discarding 20 bases for each base kept.

The condensed results of the searches performed are presented in Table 5.16. Each region was used as reference with the three target organisms and the table displays whether or not a similar sequence was found in the target. If the located region has an identified gene in the NCBI databases [80] its name is presented in the cell. When the located region does not include an identified region, only a check mark is visible. `BLASTn`, which is optimized for somewhat similar sequences, was also used to search the regions in the Nucleotide collection database and, when the region was found in the same chromosome and organism, the identity between the sequences is presented in the table.

Chromosome	Region name	Location	Length	Gene	
				Symbol	Type
3	3.1	107319672-107320868	1197	*DUBR*	ncRNA
	3.2	21683672-21684352	681	*ZNF385D*	Protein coding
	3.3	192172364-192173020	657	*FGF12*	Protein coding
4	4.1	121361770-121362434	665	*QRFPR*	Protein coding
	4.2	121363572-121364296	725		
5	5.1	133539196-133539844	649	*FSTL4*	Protein coding
6	6.1	165316348-165316952	605	*LOC105378113*	ncRNA
7	7.1	148376510-148378764	2255	*CNTNAP2*	Protein coding
	7.2	90182022-90182656	635	*STEAP2-AS1*	ncRNA
8	8.1	645828-648476	2649	*ERICH1*	Protein coding
	8.2	644534-645682	1149		
	8.3	648580-649256	677		
	8.4	40918766-40919602	837	*LOC105379389*	ncRNA
	8.5	40920586-40921294	709		
11	11.1	3218640-3219772	1133	*MRGPRG-AS1*	*ncRNA*
	11.2	3221100-3221836	737		
12	12.1	9485332-9486410	1079	*LOC105376678*	Protein coding
	12.2	9486714-9487424	711		
	12.3	9484530-9485204	675		
13	13.1	50495764-50497414	1651	*DLEU1*	ncRNA
	13.2	50497566-50498464	899		
15	15.1	25686522-25687228	707	*ATP10A*	Protein coding
	15.2	71589672-71590344	673	*THSD4*	Protein coding
18	18.1	21157810-21158606	797	*LOC107985161*	ncRNA
22	22.1	38899876-38900934	1059	*LOC105373032*	ncRNA
X	X.1	56151436-56152988	1553	*KLF8*	Protein coding
	X.2	56142228-56143754	1527		
	X.3	56149828-56150760	933		
	X.4	70721052-70721808	757	*TEX11*	Protein coding
	X.5	109658306-109658958	653	*ACSL4*	Protein coding
	X.6	109654002-109654994	993		

Table 5.15: Modern human specific regions, compared to a *Neanderthal* and a *Denisovan* genomes, used in this work. The chromosome and location refer to the modern human genome the regions belong to. Each region is referred to throughout the work with the presented name. The gene shown indicates the gene symbol and type in which the region is found.

Region name	Target organism					
	Pan troglodytes		*Gorilla gorilla*		*Pongo abelii*	
	NET-ASAR	BLAST	NET-ASAR	BLAST	NET-ASAR	BLAST
3.1	✓ *(LOC460568)*	X	✓	X	✓	X
3.2	✓ *(ZNF385D)*	X	✓ *(ZNF385D)*	X	✓ *(ZNF385D)*	X
3.3	✓ *(FGF12)*	X	✓ *(FGF12)*	X	✓ *(FGF12)*	X
4.1	✓ *(QRFPR)*	X	✓ *(QRFPR)*	X	X	X
4.2	✓ *(QRFPR)*	X	✓ *(QRFPR)*	X	✓ *(QRFPR)*	X
5.1	✓ *(FSTL4)*	X	✓ *(FSTL4)*	X	✓ *(FSTL4)*	X
6.1	✓	X	✓	X	✓	X
7.1	✓ *(CNTNAP2)*	100%	✓ *(CNTNAP2)*	X	✓ *(CNTNAP2)*	X
7.2	✓	100%	✓[2] *(STEAP1B)*	X	✓	96%
8.1	✓ *(ERICH1)*	X	✓ *(ERICH1)*	X	✓ *(LOC100445203)*	X
8.2	✓ *(ERICH1)*	90%	✓ *(ERICH1)*	X	✓ *(LOC100445203)*	X
8.3	✓ *(ERICH1)*	X	✓ *(ERICH1)*	X	✓ *(LOC100445203)*	X
8.4	✓	X	✓ *(LOC109027985)*	X	✓	X
8.5	✓	X	✓ *(LOC109027985)*	X	✓	X
11.1	✓ *(LOC100615982)*	99%	✓	96%	✓	97%
11.2	✓ *(LOC100615982)*	97%	✓	X	✓	X
12.1	✓[1] *(LOC104001461)*	X	✓	X	✓	X
12.2	✓[1]	X	✓	X	✓	X
12.3	✓[1] *(LOC104001461)*	X	✓	X	✓	X
13.1	✓	X	✓	X	✓	X
13.2	✓	X	✓	X	✓	X
15.1	✓ *(ATP10A)*	X	✓ *(ATP10A)*	X	✓ *(ATP10A)*	X
15.2	✓ *(THSD4)*	X	✓ *(THSD4)*	X	✓ *(LOC100434337)*	X
18.1	✓	X	✓	X	✓	X
22.1	✓	X	✓	X	✓ *(LOC103888973)*	X
X.1	✓ *(LOC107971107)*	X	✓	X	✓	X
X.2	✓ *(LOC107971107)*	X	✓	X	✓	X
X.3	✓ *(LOC107971107)*	X	✓	X	✓	X
X.4	✓ *(TEX11)*	99%	✓ *(TEX11)*	X	✓ *(TEX11)*	X
X.5	✓ *(ACSL4)*	99%	✓ *(ACSL4)*	X	✓ *(ACSL4)*	X
X.6	✓ *(ACSL4)*	99%	✓ *(ACSL4)*	X	✓ *(ACSL4)*	X

Table 5.16: Condensed results of the search using `NET-ASAR` with the modern human specific regions as a reference and different organisms as target. A check mark means that a similar region was located in the target. If the detected region is contained in an identified gene it is presented in the cell. The results marked with [1] and [2] refer to results with more than one identified region, with only the identified gene being presented. The search results using `BLASTn` are presented, when positive results were obtained, through the identity between the sequences.

Given the great number of plots (93) that result from the searches performed, we have opted to only focus on a few selected ones. All the plots are available at `https://github.com/manuelgaspar/NET-ASAR`.

Looking at the condensed results presented in Table 5.16, it is clear that every region was identified in the three organisms, with exception of region 4.1 in *Pongo abelii*.

Figure 5.16 presents the plot referring to the identification of the region 4.1 in chromosome 4 of *Pongo abelii*.

If one were to use a threshold of, for example, 1.8, then several regions would be identified as similar to the reference sequence. Despite this, we have found that the difference of entropy between those regions and the average entropy of the target is too small to deduce the presence of the reference, even though it reveals that some target regions have some similarity with the reference.

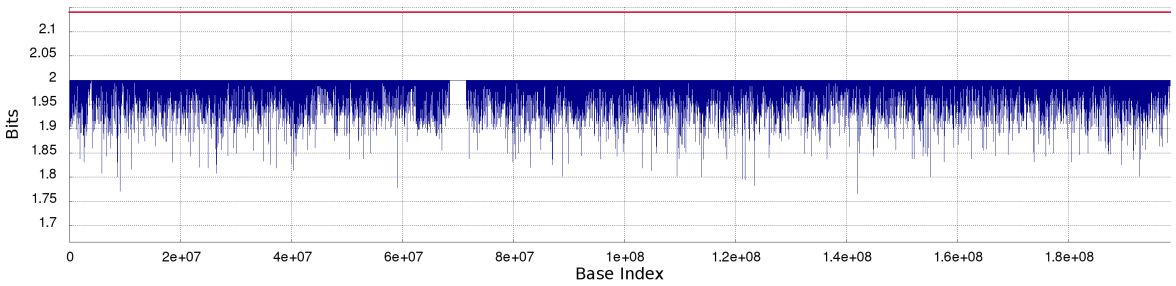
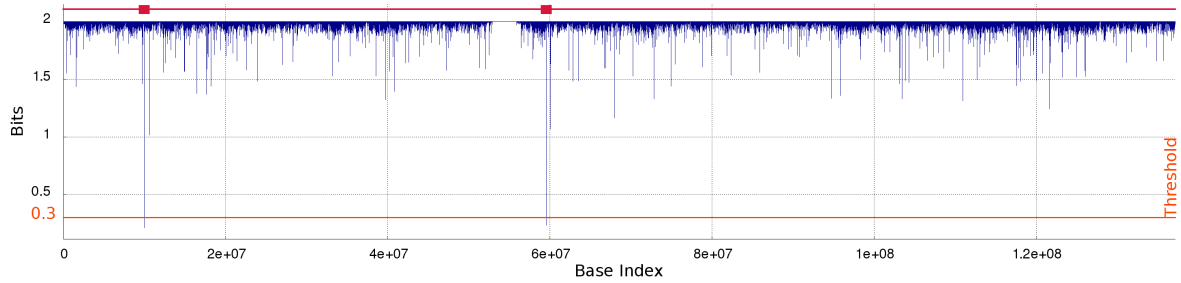


Figure 5.16: Identification of reference region 4.1 in chromosome 4 of *Pongo abelii*.

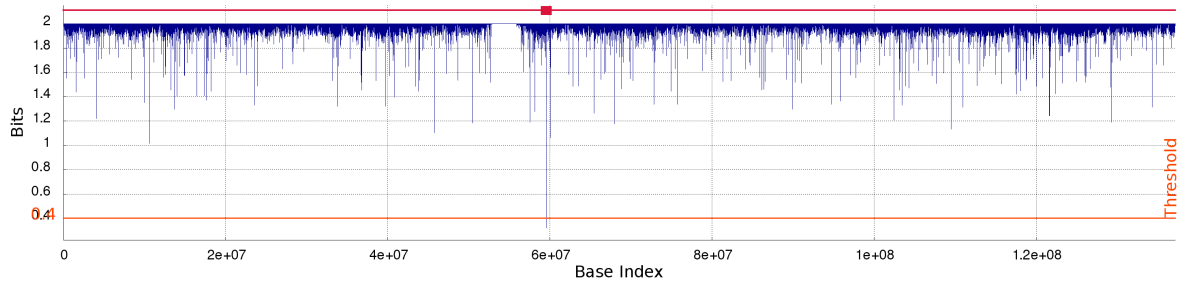
Figure 5.17 displays the plots obtained with the regions 12.1, 12.2 and 12.3 as reference and chromosome 12 of *Pan troglodytes* as target (marked in Table 5.16 with [1]). The three references were located in the target in a common location, which does not contain any identified gene, but with references 12.1 and 12.3 (Figures 5.17a and 5.17c, respectively) we located an extra region, in which the gene *LOC104001461* can be found. The assumption can be made that some degree of similarity exists between only some regions of the reference gene (*LOC105376678*) and the identified gene (*LOC104001461*), since the location is not identified when using region 12.2 as a reference.

Figure 5.18 shows the plot obtained when using region 7.2 as reference and chromosome 7 of *Gorilla gorilla* as the target (marked with [2] in Table 5.16).

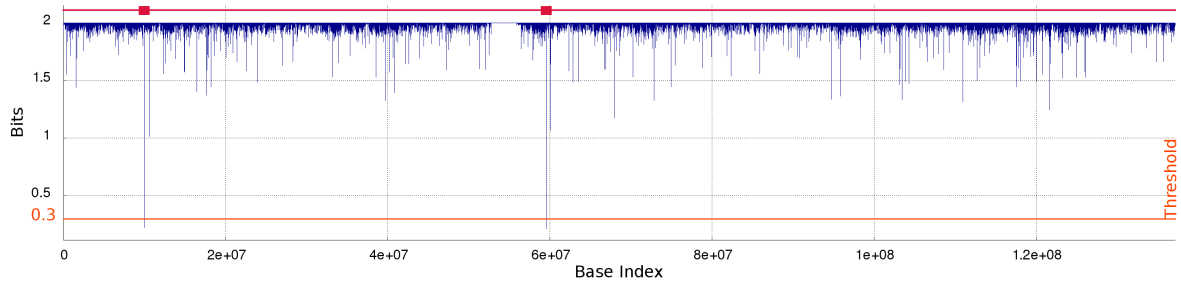
By analyzing the plot, two different highlighted regions can be seen, with the first having an identified gene (*STEAP1B*) while the second does not. As was mentioned in the previous sections, it could be a result of a low complexity region, which, in this region, is not the case. A possible explanation is that the region found has a high degree of similarity in relation to the reference region, due to the presence of a homolog. The first highlighted region, where the gene *STEAP1B* can be found, may lead to a positive result, since both genes belong to the same gene family (*STEAP*).



(a) 12.1.



(b) 12.2.



(c) 12.3.

Figure 5.17: Identification of reference regions 12.1, 12.2 and 12.3 in target chromosome 12 of *Pan troglodytes*.

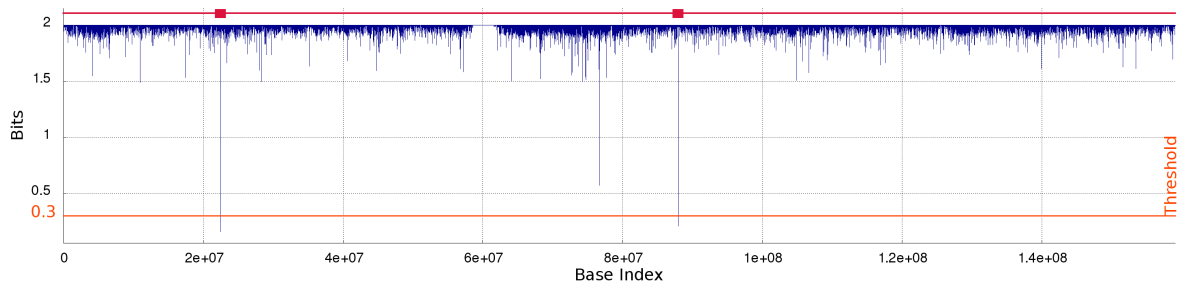


Figure 5.18: Identification of reference region 7.2 in chromosome 7 of *Gorilla gorilla*.

5.3.1 Final Remarks

If one were to only take into account the evolution tree and the known similarities between humans and remaining *Hominoidea*, it would be easy to assume that the modern human specific regions are only present in the modern human genome. However, from the searches performed in this section, we have found that the modern human specific regions were present in three close species (*Pan troglodytes*, *Gorilla gorilla* and *Pongo abelii*).

The potential of `NET-ASAR` is also apparent, since it was able to identify most of the regions, while the `BLAST` algorithm was only able to locate the regions in the database in 11 of the 93 searches.

Due to the somewhat unexpected nature of the results obtained in this section, we feel that further studies are warranted.

Although it falls slightly out of scope of this thesis, we propose three possible justifications for the obtained results:

- The regions can have high genomic variation, which can contain substantial postmortem degradation [101] and high substitution mutation rates (caused by PCR amplification) [102].
- Throughout time, it is possible that microorganisms affect the host macro organism, so, considering that the genomes are from organisms that did not coexist (the reference genomes are from present days specimens), they might have been affected differently [103]. External factors, such as the introduction of agriculture, can also have an effect [104].
- There is genetic variation between individuals of the same species [105, 106], and, as such, the fact that the ancient DNA was sequenced from a single individual, leads to low diversity in the sequenced genomes. On the contrary, the reference genomes were sequenced from specimens found today, while being sequenced from several individuals, making them a better model of the whole species genome.

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

The research presented in this thesis had as main goal the development of an alignment-free technique to locate DNA subsequences with some degree of similarity, relative to a larger target sequence, using Finite-Context Models.

By taking advantage of some characteristics of DNA, the method, based on data compression, was implemented and tested under different conditions, providing positive outcomes.

One of the facts to take into account is that, while the method can provide correct results, it also demands a high degree of familiarity with the method's usage. Since the process cannot be completely automated, the user is required to determine the correct parameters, such as context size and filtering, needed for each search. The choice must be made having in mind the characteristics of both the reference and target sequences. It should also be noted that, even though the results of some given search can be satisfactory, the user should also analyze the identified regions closely, as false positives are a possibility.

We have located modern human specific regions, with relation to a *Neanderthal* and a *Denisovan* high coverage genomes, in the genomes of different closely related species of the modern human, using the developed tool, which also serves as a proof of concept for possible scientific applications.

In regards to the application with ancient DNA, we consider that further study is required to fully understand the justification for the obtained results. Since that study was not the main objective of this work we have only presented three possible justifications (although not supported by deep examination).

In future work, regarding the tool, it would be beneficial to improve the algorithm, as it is demanding in terms of execution time and computational resources (when comparing to some methods). Furthermore, it could be developed a way of automating the tool, in order to easily adapt to different conditions, by selecting the more suitable parameters automatically.

BIBLIOGRAPHY

[1] Stéphane Grumbach and Fariza Tahi. A new challenge for compression algorithms: Genetic sequences. *Information Processing & Management*, 30(6):875–886, 1994.

[2] Xin Chen, Ming Li, Bin Ma, and John Tromp. DNACompress: fast and effective DNA sequence compression. *Bioinformatics*, 18(12):1696–1698, 2002.

[3] Gergely Korodi and Ioan Tabus. An efficient normalized maximum likelihood algorithm for DNA sequence compression. *ACM Transactions on Information Systems (TOIS)*, 23(1):3–34, 2005.

[4] Ashutosh Gupta and Suneeta Agarwal. A novel approach for compressing DNA sequences using semi-statistical compressor. *International Journal of Computers and Applications*, 33(3):245–251, 2011.

[5] Armando J Pinho, Paulo JSG Ferreira, António JR Neves, and Carlos AC Bastos. On the representability of complete genomes by multiple competing finite-context (Markov) models. *PLoS ONE*, 6(6):e21588, 2011.

[6] Minh Duc Cao, Trevor I Dix, Lloyd Allison, and Chris Mears. A simple statistical algorithm for biological sequence compression. In *Data Compression Conference, 2007. DCC'07*, pages 43–52. IEEE, 2007.

[7] Scott Christley, Yiming Lu, Chen Li, and Xiaohui Xie. Human genomes as email attachments. *Bioinformatics*, 25(2):274–275, 2008.

[8] Congmao Wang and Dabing Zhang. A novel compression tool for efficient storage of genome resequencing data. *Nucleic Acids Research*, 39(7):e45–e45, 2011.

[9] Sebastian Deorowicz, Agnieszka Danek, and Marcin Niemiec. GDC 2: Compression of large collections of genomes. *Scientific Reports*, 5, 2015.

[10] Shanika Kuruppu, Bryan Beresford-Smith, Thomas Conway, and Justin Zobel. Iterative dictionary construction for compression of large DNA data sets. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 9(1):137–149, 2012.

[11] Stephen F Altschul, Thomas L Madden, Alejandro A Schäffer, Jinghui Zhang, Zheng Zhang, Webb Miller, and David J Lipman. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research*, 25(17):3389–3402, 1997.

[12] Julie D Thompson, Desmond G Higgins, and Toby J Gibson. CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic acids research*, 22(22):4673–4680, 1994.

[13] Robert C Edgar. MUSCLE: multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Research*, 32(5):1792–1797, 2004.

[14] Robert D Finn, Alex Bateman, Jody Clements, Penelope Coggill, Ruth Y Eberhardt, Sean R Eddy, Andreas Heger, Kirstie Hetherington, Liisa Holm, Jaina Mistry, et al. Pfam: the protein families database. *Nucleic Acids Research*, 42(D1):D222–D230, 2013.

[15] Aaron E Darling, Bob Mau, and Nicole T Perna. progressiveMauve: multiple genome alignment with gene gain, loss and rearrangement. *PLoS ONE*, 5(6):e11147, 2010.

[16] Ricardo A Vialle, Fabio O Pedrosa, Vinicius A Weiss, Dieval Guizelini, Juliana H Tibaes, Jeroniza N Marchaukoski, Emanuel M de Souza, and Roberto T Raittz. RAFTS3: Rapid Alignment-Free Tool for Sequence Similarity Search. *bioRxiv*, page 055269, 2016.

[17] Sebastian Horwege, Sebastian Lindner, Marcus Boden, Klas Hatje, Martin Kollmar, Chris-André Leimeister, and Burkhard Morgenstern. Spaced words and kmacs: fast alignment-free sequence comparison based on inexact word matches. *Nucleic Acids Research*, 42(W1):W7–W11, 2014.

[18] Andrzej Zielezinski, Susana Vinga, Jonas Almeida, and Wojciech M Karlowski. Alignment-free sequence comparison: benefits, applications, and tools. *Genome Biology*, 18(1):186, 2017.

[19] Carl R Woese and George E Fox. Phylogenetic structure of the prokaryotic domain: the primary kingdoms. *Proceedings of the National Academy of Sciences*, 74(11):5088–5090, 1977.

[20] Eugene V Koonin, Tatiana G Senkevich, and Valerian V Dolja. The ancient Virus World and evolution of cells. *Biology Direct*, 1(1):29, 2006.

[21] [the genome of the brown alga ectocarpus siliculosus contains a series of viral dna pieces, suggesting an ancient association with large dsdna viruses.

[22] Florian Maumus, Aline Epert, Fabien Nogué, and Guillaume Blanc. Plant genomes enclose footprints of past infections by giant virus relatives. *Nature Communications*, 5, 2014.

[23] Jonathan Filée. Multiple occurrences of giant virus core genes acquired by eukaryotic genomes: The visible part of the iceberg? *Virology*, 466:53–59, 2014.

[24] Colson, Philippe and De Lamballerie, Xavier and Yutin, Natalya and Asgari, Sassan and Bigot, Yves and Bideshi, Dennis K and Cheng, Xiao-Wen and Federici, Brian A and Van Etten, James L and Koonin, Eugene V and others. "Megavirales", a proposed new order for eukaryotic nucleocytoplasmic large DNA viruses. *Archives of Virology*, 158(12):2517–2521, 2013.

[25] Patrick Forterre, Mart Krupovic, and David Prangishvili. Cellular domains and viral lineages. *Trends in Microbiology*, 22(10):554–558, 2014.

[26] Elizabeth Pennisi. Ever-bigger viruses shake tree of life. *Science*, 341(6143):226–227, 2013.

[27] Carlos Canchaya, Ghislain Fournous, Sandra Chibani-Chennoufi, Marie-Lise Dillmann, and Harald Brüssow. Phage as agents of lateral gene transfer. *Current Opinion in Microbiology*, 6(4):417–424, 2003.

[28] James D Watson, Francis HC Crick, et al. Molecular structure of Nucleic Acids. *Nature*, 171(4356):737–738, 1953.

[29] Frederick Sanger, Steven Nicklen, and Alan R Coulson. DNA sequencing with chain-terminating inhibitors. *Proceedings of the National Academy of Sciences*, 74(12):5463–5467, 1977.

[30] Walter Gilbert. Why genes in pieces? *Nature*, 271(5645):501–501, 1978.

[31] From DNA to RNA to protein, how does it work? `https://science-explained.com/theory/dna-rna-and-protein/`. (Accessed on 21/09/2017).

[32] Francis HC Crick, Leslie Barnett, Sydney Brenner, and Richard J Watts-Tobin. General nature of the genetic code for proteins. *Nature*, 192(4809):1227–1232, 1961.

[33] J Tomkins. How genomes are sequenced and why it matters: Implications for studies in comparative genomics of humans and chimpanzees. *Answers Research Journal*, 4:81–88, 2011.

[34] Nikolay V Dokholyan, Sergey V Buldyrev, Shlomo Havlin, and H Eugene Stanley. Distribution of base pair repeats in coding and noncoding DNA sequences. *Physical Review Letters*, 79(25):5182, 1997.

[35] Edward N Trifonov and Joel L Sussman. The pitch of chromatin DNA is reflected in its nucleotide sequence. *Proceedings of the National Academy of Sciences*, 77(7):3816–3820, 1980.

[36] Joaquín Sánchez and Imelda Lopez-Villasenor. A simple model to explain three-base periodicity in coding DNA. *FEBS letters*, 580(27):6413–6422, 2006.

[37] Armando J Pinho, António JR Neves, Vera Afreixo, Carlos AC Bastos, and Paulo Jorge SG Ferreira. A three-state model for DNA protein-coding regions. *IEEE Transactions on Biomedical Engineering*, 53(11):2148–2155, 2006.

[38] Claude E Shannon. A mathematical theory of communication. *Bell Syst. Tech. J.*, 27:379–423,623–656, 1948.

[39] David A Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101, 1952.

[40] Solomon Golomb. Run-length encodings (Corresp.). *IEEE Transactions on Information Theory*, 12(3):399–401, 1966.

[41] Khalid Sayood. *Introduction to Data Compression*. Newnes, 2012.

[42] L Peter Deutsch. DEFLATE compressed data format specification version 1.3. 1996.

[43] Gregory K Wallace. The JPEG still picture compression standard. *IEEE Transactions on Consumer Electronics*, 38(1):xviii–xxxiv, 1992.

[44] Jacob Ziv and Abraham Lempel. A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, 23(3):337–343, 1977.

[45] Jacob Ziv and Abraham Lempel. Compression of individual sequences via variable-rate coding. *IEEE Transactions on Information Theory*, 24(5):530–536, 1978.

[46] Terry A. Welch. A technique for high-performance data compression. *Computer*, 6(17):8–19, 1984.

[47] Jorma Rissanen and Glen G Langdon. Arithmetic coding. *IBM Journal of research and development*, 23(2):149–162, 1979.

[48] Jorma Rissanen and Glen G Langdon. Universal modeling and coding. *IEEE Transactions on Information Theory*, 27(1):12–23, 1981.

[49] Ian H Witten, Radford M Neal, and John G Cleary. Arithmetic coding for data compression. *Communications of the ACM*, 30(6):520–540, 1987.

[50] D. Salomon. *Data compression - The complete reference*. Springer, 2 edition, 2000.

[51] Timothy C Bell, John G Cleary, and Ian Witten. *Text Compression*. Prentice Hall, 1990.

[52] Armando J Pinho, António JR Neves, Carlos AC Bastos, Daniel A Martins, and Paulo JSG Ferreira. *Finite-context models for DNA coding*. INTECH Open Access Publisher, 2010.

[53] Linda Stern, Lloyd Allison, Ross L Coppel, and Trevor I Dix. Discovering patterns in Plasmodium falciparum genomic DNA. *Molecular and Biochemical Parasitology*, 118(2):175–186, 2001.

[54] Xin Chen, Sam Kwong, and Ming Li. A compression algorithm for DNA sequences and its applications in genome comparison. *Genome Informatics*, 10:51–61, 1999.

[55] David R Powell, Lloyd Allison, and Trevor I Dix. Modelling-Alignment for Non-random Sequences. In *Australian Conference on Artificial Intelligence*, pages 203–214. Springer, 2004.

[56] Sebastian Deorowicz and Szymon Grabowski. Data compression for sequencing data. *Algorithms for Molecular Biology*, 8(1):25, 2013.

[57] Morteza Hosseini, Diogo Pratas, and Armando J Pinho. A survey on data compression methods for biological sequences. *Information*, 7(4):56, 2016.

[58] Diogo Pratas, Armando J Pinho, and Paulo JSG Ferreira. Efficient compression of genomic sequences. In *Data Compression Conference (DCC), 2016*, pages 231–240. IEEE, 2016.

[59] Eric S Lander. Initial impact of the sequencing of the human genome. *Nature*, 470(7333):187, 2011.

[60] Markus Hsi-Yang Fritz, Rasko Leinonen, Guy Cochrane, and Ewan Birney. Efficient storage of high throughput DNA sequencing data using reference-based compression. *Genome Research*, 21(5):734–740, 2011.

[61] Zexuan Zhu, Yongpeng Zhang, Zhen Ji, Shan He, and Xiao Yang. High-throughput DNA sequence data compression. *Briefings in Bioinformatics*, 16(1):1–15, 2013.

[62] Armando J Pinho, Diogo Pratas, and Sara P Garcia. GReEn: a tool for efficient compression of genome resequencing data. *Nucleic Acids Research*, 40(4):e27–e27, 2011.

[63] Idoia Ochoa, Mikel Hernaez, and Tsachy Weissman. iDoComp: a compression scheme for assembled genomes. *Bioinformatics*, 31(5):626–633, 2014.

[64] Sebastian Deorowicz and Szymon Grabowski. Robust relative compression of genomes with random access. *Bioinformatics*, 27(21):2979–2986, 2011.

[65] James A Storer and Thomas G Szymanski. Data compression via textual substitution. *Journal of the ACM (JACM)*, 29(4):928–951, 1982.

[66] AN Kolmogorov. Three approaches to the quantitative definition of information. *Problems of Information Transmission*, 1(1):1–7, 1965.

[67] Alan Mathison Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 2(1):230–265, 1937.

[68] Trevor I Dix, David R Powell, Lloyd Allison, Julie Bernal, Samira Jaeger, and Linda Stern. Comparative analysis of long DNA sequences by per element information content using different contexts. *BMC Bioinformatics*, 8(2):S10, 2007.

[69] Abraham Lempel and Jacob Ziv. On the complexity of finite sequences. *IEEE Transactions on Information Theory*, 22(1):75–81, 1976.

[70] Ming Li, Xin Chen, Xin Li, Bin Ma, and Paul MB Vitányi. The similarity metric. *IEEE Transactions on Information Theory*, 50(12):3250–3264, 2004.

[71] Charles H Bennett, Péter Gács, Ming Li, Paul MB Vitányi, and Wojciech H Zurek. Information distance. *IEEE Transactions on Information Theory*, 44(4):1407–1423, 1998.

[72] Rudi Cilibrasi and Paul MB Vitányi. Clustering by compression. *IEEE Transactions on Information Theory*, 51(4):1523–1545, 2005.

[73] Diogo Pratas, Raquel M Silva, Armando J Pinho, and Paulo JSG Ferreira. An alignment-free method to find and visualise rearrangements between pairs of DNA sequences.

[74] Fei Nan and Donald Adjeroh. On complexity measures for biological sequences. In *Computational Systems Bioinformatics Conference, 2004. CSB 2004. Proceedings. 2004 IEEE*, pages 522–526. IEEE, 2004.

[75] Leila Pirhaji, Mehdi Kargar, Armita Sheari, Hadi Poormohammadi, Mehdi Sadeghi, Hamid Pezeshk, and Changiz Eslahchi. The performances of the chi-square test and complexity measures for signal recognition in biological sequences. *Journal of Theoretical Biology*, 251(2):380–387, 2008.

[76] [authorship attribution using relative compression.

[77] Diogo Pratas. pratas/goose v1.1. Zenodo. https://doi.org/10.5281/zenodo.1009142, October 2017.

[78] Stephen F Altschul, Warren Gish, Webb Miller, Eugene W Myers, and David J Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403–410, 1990.

[79] Tatiana A Tatusova and Thomas L Madden. BLAST 2 Sequences, a new tool for comparing protein and nucleotide sequences. *FEMS Microbiology Letters*, 174(2):247–250, 1999.

[80] Resource Coordinators NCBI. Database Resources of the National Center for Biotechnology Information. *Nucleic Acids Research*, 45(D1):D12, 2017.

[81] Diogo Pratas, Morteza Hosseini, Raquel M Silva, Armando J Pinho, and Paulo JSG Ferreira. Visualization of Distinct DNA Regions of the Modern Human Relatively to a Neanderthal Genome. In *Iberian Conference on Pattern Recognition and Image Analysis*, pages 235–242. Springer, 2017.

[82] Diogo Pratas, Raquel M Silva, Armando J Pinho, and Paulo JSG Ferreira. Detection and visualisation of regions of human DNA not present in other primates. *Proceedings of the 21st Portuguese Conference on Pattern Recognition, RecPad*, 2015.

[83] Kay Prüfer, Fernando Racimo, Nick Patterson, Flora Jay, Sriram Sankararaman, Susanna Sawyer, Anja Heinze, Gabriel Renaud, Peter H Sudmant, Cesare De Filippo, et al. The complete genome sequence of a Neanderthal from the Altai Mountains. *Nature*, 505(7481):43–49, 2014.

[84] Matthias Meyer, Martin Kircher, Marie-Theres Gansauge, Heng Li, Fernando Racimo, Swapan Mallick, Joshua G Schraiber, Flora Jay, Kay Prüfer, Cesare De Filippo, et al. A high-coverage genome sequence from an archaic Denisovan individual. *Science*, 338(6104):222–226, 2012.

[85] John D Eicher, Natalie R Powers, Laura L Miller, Natacha Akshoomoff, David G Amaral, Cinnamon S Bloss, Ondrej Libiger, Nicholas J Schork, Burcu F Darst, BJ Casey, et al. Genome-wide association study of shared components of reading disability and language impairment. *Genes, Brain and Behavior*, 12(8):792–801, 2013.

[86] Ashish Bhushan, Avninder Singh, Sujala Kapur, Bibhuti B Borthakar, Jagannath Sharma, Avdhesh K Rai, Amal C Kataki, and Sunita Saxena. Identification and Validation of Fibroblast Growth Factor 12 Gene as a Novel Potential Biomarker in Esophageal Cancer Using Cancer Genomic Datasets. *OMICS: A Journal of Integrative Biology*, 21(10):616–631, 2017.

[87] Claus-Jürgen Scholz, Heike Weber, Susanne Jungwirth, Walter Danielczyk, Andreas Reif, Karl-Heinz Tragl, Peter Fischer, Peter Riederer, Jürgen Deckert, and Edna Grünblatt. Explorative results from multistep screening for potential genetic risk loci of Alzheimers disease in the longitudinal VITA study cohort. *Journal of Neural Transmission*, pages 1–11, 2017.

[88] May M Luke, Ellen S O'meara, Charles M Rowland, Dov Shiffman, Lance A Bare, Andre R Arellano, WT Longstreth, Thomas Lumley, Kenneth Rice, Russell P Tracy, et al. Gene variants associated with ischemic stroke. *Stroke*, 40(2):363–368, 2009.

[89] Erin Flaherty, Rania M Deranieh, Elena Artimovich, Inkyu S Lee, Arthur J Siegel, Deborah L Levy, Michael W Nestor, and Kristen J Brennand. Patient-derived hiPSC neurons with heterozygous CNTNAP2 deletions display altered neuronal gene expression and network activity. *npj Schizophrenia*, 3(1):35, 2017.

[90] Daniel Vogt, Kathleen KA Cho, Samantha M Shelton, Anirban Paul, Z Josh Huang, Vikaas S Sohal, and John LR Rubenstein. Mouse Cntnap2 and Human CNTNAP2 ASD Alleles Cell Autonomously Regulate PV+ Cortical Interneurons. *Cerebral Cortex*, pages 1–12, 2017.

[91] Helen Whiteland, Samantha Spencer-Harty, Claire Morgan, Howard Kynaston, David Hywel Thomas, Pradeep Bose, Neil Fenn, Paul Lewis, Spencer Jenkins, and Shareen H Doak. A role for STEAP2 in prostate cancer progression. *Clinical & Experimental Metastasis*, 31(8):909–920, 2014.

[92] Tomohiko Harada, Claude Chelala, Tatjana Crnogorac-Jurcevic, and Nicholas R Lemoine. Genome-wide analysis of pancreatic cancer using microarray-based techniques. *Pancreatology*, 9(1-2):13–24, 2009.

[93] Li-Li Wang, Kai-Xuan Sun, Dan-Dan Wu, Yin-Ling Xiu, Xi Chen, Shuo Chen, Zhi-Hong Zong, Xiu-Bo Sang, Yao Liu, and Yang Zhao. DLEU1 contributes to ovarian carcinoma tumourigenesis and development by interacting with miR-490-3p and altering CDK1 expression. *Journal of Cellular and Molecular Medicine*, 2017.

[94] Sanghoon Lee, Wen Luo, Tishi Shah, Changhong Yin, Timmy OConnell, Tae-Hoon Chung, Sherrie L Perkins, Rodney R Miles, Janet Ayello, Erin Morris, et al. The effects of DLEU1 gene expression in Burkitt lymphoma (BL): potential mechanism of chemoimmunotherapy resistance in BL. *Oncotarget*, 8(17):27839, 2017.

[95] Tomoki Naito, Hiroyuki Takatsu, Rie Miyano, Naoto Takada, Kazuhisa Nakayama, and Hye-Won Shin. Phospholipid flippase ATP10A translocates phosphatidylcholine and is involved in plasma membrane dynamics. *Journal of Biological Chemistry*, 290(24):15004–15017, 2015.

[96] Peng Su, Shiwang Wen, Yuefeng Zhang, Yong Li, Yanzhao Xu, Yonggang Zhu, Huilai Lv, Fan Zhang, Mingbo Wang, and Ziqiang Tian. Identification of the key genes and pathways in esophageal carcinoma. *Gastroenterology Research and Practice*, 2016, 2016.

[97] Jianchao Li, Yifei Liu, Jianhua Xue, Mingming Xu, Jianguo Zhang, Junhua Liu, and Wenyi Wang. Krüppel-Like Factor 8 Overexpression Correlates with Poor Prognosis in Non-Small Cell Lung Cancer. *Pathology & Oncology Research*, pages 1–7, 2017.

[98] Xiaoping Yi, Hongyan Zai, Xueying Long, Xiaoyi Wang, Wenzheng Li, and Yixiong Li. Krüppel-like factor 8 induces epithelial-to-mesenchymal transition and promotes invasion of pancreatic cancer cells through transcriptional activation of four and a half LIM-only protein 2. *Oncology Letters*, 14(4):4883–4889, 2017.

[99] Fang Yang, Sherman Silber, N Adrian Leu, Robert D Oates, Janet D Marszalek, Helen Skaletsky, Laura G Brown, Steve Rozen, David C Page, and P Jeremy Wang. TEX11 is mutated in infertile men with azoospermia and regulates genome-wide recombination rates in mouse. *EMBO Molecular Medicine*, 7(9):1198–1210, 2015.

[100] Xiao-Jie Sun and Ge-Liang Xu. Overexpression of Acyl-CoA Ligase 4 (ACSL4) in Patients with Hepatocellular Carcinoma and its Prognosis. *Medical Science Monitor: International Medical Journal of Experimental and Clinical Research*, 23:4343, 2017.

[101] Walter Bär, Adelgunde Kratzer, Marco Mächler, and Werner Schmid. Postmortem stability of DNA. *Forensic Science International*, 39(1):59–70, 1988.

[102] Elsje Pienaar, M Theron, Michael Nelson, and Hendrik J Viljoen. A quantitative model of error accumulation during PCR amplification. *Computational Biology and Chemistry*, 30(2):102–111, 2006.

[103] Ravi Jain, Maria C Rivera, and James A Lake. Horizontal gene transfer among genomes: the complexity hypothesis. *Proceedings of the National Academy of Sciences*, 96(7):3801–3806, 1999.

[104] Iain Mathieson, Iosif Lazaridis, Nadin Rohland, Swapan Mallick, Nick Patterson, Songül Alpaslan Roodenberg, Eadaoin Harney, Kristin Stewardson, Daniel Fernandes, Mario Novak, et al. Genome-wide patterns of selection in 230 ancient Eurasians. *Nature*, 528(7583):499–503, 2015.

[105] Ravi Sachidanandam, David Weissman, Steven C Schmidt, Jerzy M Kakol, Lincoln D Stein, Gabor Marth, Steve Sherry, James C Mullikin, Beverley J Mortimore, David L Willey, et al. A map of human genome sequence variation containing 1.42 million single nucleotide polymorphisms. *Nature*, 409(6822):928–933, 2001.

[106] Kenneth K Kidd, AJ Pakstis, WC Speed, and JR Kidd. Understanding human DNA sequence variation. *Journal of Heredity*, 95(5):406–420, 2004.