Universidade de Aveiro
2014

Departamento de
Electrónica, Telecomunicações e Informática
Programa Doutoral em Ciências da Computação
Universidades do Minho, Aveiro e do Porto

Paulo Miguel da Silva Gaspar

Métodos computacionais para a caracterização de genes e extração de conhecimento genómico

Computational methods for gene characterization and genomic knowledge extraction

Universidade de Aveiro
2014

Departamento de
Electrónica, Telecomunicações e Informática
Programa Doutoral em Ciências da Computação
Universidades do Minho, Aveiro e do Porto

Paulo Miguel da Silva Gaspar

Métodos computacionais para a caracterização de genes e extração de conhecimento genómico

Computational methods for gene characterization and genomic knowledge extraction

Tese apresentada às Universidades do Minho, Aveiro e Porto para cumprimento dos requisitos necessários à obtenção do grau de Doutor no âmbito do doutoramento conjunto MAP-i, realizada sob a orientação científica do Doutor José Luís Guimarães Oliveira, Professor Associado do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro, e do Doutor Sérgio Guilherme Aleixo de Matos, Investigador Auxiliar do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro.

o júri / the juri

presidente

Doutor Fernando Joaquim Fernandes Tavares Rocha
Professor Catedrático do Departamento de Geociências da Universidade de Aveiro

Doutor Fernando Manuel Augusto da Silva
Professor Catedrático da Faculdade de Ciências da Universidade do Porto

Doutora Ana Teresa Correia de Freitas
Professora Associada com Agregação do Departamento de Engenharia Informática da Universidade de Lisboa

José Luís Guimarães Oliveira
Professor Associado do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro (orientador)

Doutor Rui Pedro Sanches de Castro Lopes
Professor Coordenador da Escola Superior de Tecnologia e de Gestão do Instituto Politécnico de Bragança

Doutor Joel Perdiz Arrais
Professor Auxiliar Convidado da Faculdade de Ciências e Tecnologia da Universidade de Coimbra

palavras-chave

bioinformática, biologia computacional, optimização, aprendizagem máquina, varioma, gene, genoma, previsão de patogenicidade, mineração de dados

resumo

Motivação: A medicina e as ciências da saúde estão atualmente num processo de alteração que muda o paradigma clássico baseado em sintomas para um personalizado e baseado na genética. O valor do impacto desta mudança nos cuidados da saúde é inestimável. Não obstante as contribuições dos avanços na genética para o conhecimento do organismo humano até agora, as descobertas realizadas recentemente por algumas iniciativas forneceram uma caracterização detalhada das diferenças genéticas humanas, abrindo o caminho a uma nova era de diagnóstico médico e medicina personalizada.
Os dados gerados por estas e outras iniciativas estão disponíveis mas o seu volume está muito para lá do humanamente explorável, e é portanto da responsabilidade dos cientistas informáticos criar os meios para extrair a informação e conhecimento contidos nesses dados.
Dentro dos dados disponíveis estão estruturas genéticas que contêm uma quantidade significativa de informação codificada que tem vindo a ser descoberta nas últimas décadas. Encontrar, ler e interpretar essa informação são passos necessários para construir modelos computacionais de entidades genéticas, organismos e doenças; uma meta que, em devido tempo, leva a benefícios humanos.

Objetivos: É possível encontrar vários padrões no varioma e exoma humano. Explorar estes padrões permite a análise e manipulação computacional de dados genéticos digitais, mas requer algoritmos especializados. Neste trabalho procurámos criar e explorar metodologias eficientes para o cálculo e combinação de padrões biológicos conhecidos, com a intenção de realizar otimizações *in silico* de estruturas genéticas, análises de genes humanos, e previsão da patogenicidade a partir de diferenças genéticas humanas.

Resultados: Concebemos várias estratégias computacionais para avaliar genes, explorar genomas, manipular sequências, e analisar o varioma de pacientes. Recorrendo a técnicas combinatórias e de otimização criámos e conjugámos algoritmos de redesenho de sequências para controlar estruturas genéticas; através da combinação do acesso a vários *web-services* e recursos externos criámos ferramentas para explorar e analisar dados genéticos, incluindo dados de pacientes; e através da aprendizagem automática desenvolvemos um procedimento para analisar mutações humanas e prever a sua patogenicidade.

keywords

bioinformatics, computational biology, optimization, machine learning, variome, gene, genome, pathogenicity prediction, data mining

abstract

Motivation: Medicine and health sciences are changing from the classical symptom-based to a more personalized and genetics-based paradigm, with an invaluable impact in health-care. While advancements in genetics were already contributing significantly to the knowledge of the human organism, the breakthrough achieved by several recent initiatives provided a comprehensive characterization of the human genetic differences, paving the way for a new era of medical diagnosis and personalized medicine.
Data generated from these and posterior experiments are now becoming available, but its volume is now well over the humanly feasible to explore. It is then the responsibility of computer scientists to create the means for extracting the information and knowledge contained in that data.
Within the available data, genetic structures contain significant amounts of encoded information that has been uncovered in the past decades. Finding, reading and interpreting that information are necessary steps for building computational models of genetic entities, organisms and diseases; a goal that in due course leads to human benefits.

Aims: Numerous patterns can be found within the human variome and exome. Exploring these patterns enables the computational analysis and manipulation of digital genomic data, but requires specialized algorithmic approaches. In this work we sought to create and explore efficient methodologies to computationally calculate and combine known biological patterns for various purposes, such as the in silico optimization of genetic structures, analysis of human genes, and prediction of pathogenicity from human genetic variants.

Results: We devised several computational strategies to evaluate genes, explore genomes, manipulate sequences, and analyze patients' variomes. By resorting to combinatorial and optimization techniques we were able to create and combine sequence redesign algorithms to control genetic structures; by combining the access to several web-services and external resources we created tools to explore and analyze available genetic data and patient data; and by using machine learning we developed a workflow for analyzing human mutations and predicting their pathogenicity.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 | INTRODUCTION

1.1 GENETICS AS A NEW HEALTHCARE PARADIGM

In 1953 James Watson and Francis Crick announced to the world the structure of DNA, a double helix resembling a twisted ladder (Figure 1.1) [193]. At the time, it had already been shown that the DNA was in fact genetic material[1], the foundation of heredity, and that it was responsible for templating the construction of proteins, with protein themselves having a crucial role in organisms. Five years later Crick introduced an hypothesis dealing with the subject of protein biosynthesis, the process by which proteins are created inside cells [47]. The central dogma of molecular biology, as it was called then, explains how information coded in DNA is transformed to RNA and then to proteins.

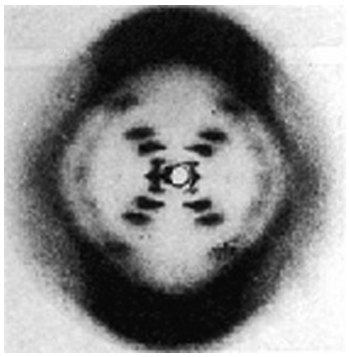


Figure 1.1 X-ray diffraction photography of a DNA molecule, taken in 1953 by Rosalind Franklin. It led Watson and Crick to the discovery of the double helix shape.

But more than a central role in building proteins, that hypothesis held the building blocks to answer a wide variety of questions, mainly regarding the mechanisms for the etiology of differentiation and disease. That hypothesis became so central that much of current molecular biology revolves around it, and several important branches have emerged from it since, including computational biology and genetic mutations research, the themes of this work.

Granted, Crick and Watson paved the way for an unprecedented evolution in the comprehension of living beings. In the past 15 years alone, we have witnessed major breakthroughs in the understanding of genetics, such as the first complete sequencing of the human genome in 2003 [45] which, for science, brought novel insights onto the architecture and function of genes and diseases [127],

[1]Since 1944, with the popular Avery-MacLeod-McCarty experiment [13].

but also encouraged numerous health-related companies to explore the released data to create diagnosing solutions, such as genetic tests for breast cancer. The 1000 genomes project followed the lead in 2008 by multiplying the experiment into a thousand and obtaining a comprehensive overview of the differences and similarities among human beings [2]. The result is an integrated map of genetic variation, detailing the profiles of individuals from different geographic locations and origins, with millions of mutations.

Understanding the implications of a genetic alteration and being able to control those changes could easily bring to science and society a breakthrough as significant as Crick and Watson's initial discovery, by changing the way medicine and healthcare behave. Furthermore, though taking control over the genes of an organism is still an early-stage idea, understanding genetic consequence is an active field of research with tangible results. As an example, since 2008 about 1800 genome wide association studies (commonly known as GWAS) have been made for more than 1000 diseases, medication susceptibilities, and other traits ranging from physical phenotypes (e.g. weight, hair color) to cognitive capabilities. In these studies, the genomes of small populations of individuals with a common target trait are compared, and the similarities among them and differences to control groups allow identifying the genetic causes for those traits. Needless to mention how this approach streamlines the creation of treatments and medicines [115]. As a consequence, genetics is rapidly becoming the cornerstone of pathology and epidemiology [127].

The amount and complexity of the genomic data made available with these and other projects far surpasses the analytical capabilities of humans, restating the role of computer science and biotechnology. Following the need for data manipulation and knowledge extraction tools, computational biology took the place as the main auxiliary discipline to help driving biology. An example of computational requirements lies with the sequenced human genome itself, which bears more than 3 billion nucleotide bases, and for which searches, annotations and alignments are intricate procedures. Computer applications such as BLAST [88] are now widely used on a daily basis to look for sequences, not only in the human genome, but also in the 655 billion nucleotide bases of 280 thousand formally described species[2] available in databases such as GenBank [19] (see Figure 1.2). Another, more recent, example comes from the spawn of variation data coming from several projects, such as the 1000 genomes. The massive amount of data made available regarding human mutations opens an

[2]With about 3800 new species added every month.

unprecedented opportunity to gather knowledge on the mechanisms of diseases and other phenotypes. Dealing with such bits of information is only now gaining wide popularity.

Furthermore, it has been pinpointed that some of the most important challenges faced by bioinformatics include the personalized interpretation of a human genome, along with the functional effects and impact of genomic variation [35, 58]. This goal is intrinsically related with a moving healthcare paradigm, leaning towards a more customized and less dogmatic medicine, increasingly relying on genetics. Although several computational and technological solutions have already emerged from research [139], we are still far from the ideal patient diagnosing pipeline: sequencing a patient genome prematurely; analyzing DNA *in silico*; and inferring pathological issues in advance to their manifestation.

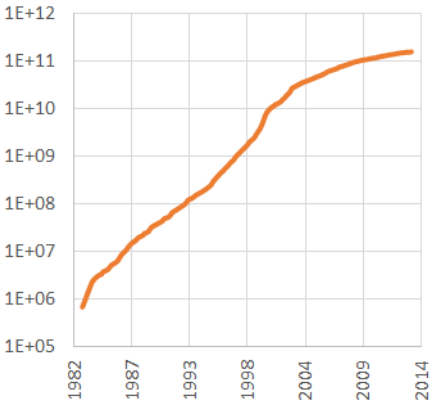


Figure 1.2 Number of DNA bases in GenBank over the years, in log scale.

Discovering the underlying information of genes, describing protein functionality and interactions, and assessing the effect and significance of the genetic information for cells are some of the most important tasks in the fields of both Molecular and Computational Biology. In this subject, several major discoveries have been made in recent years, allowing for a more comprehensive understanding of the factors and mechanisms involved in cell functioning, specially in the topic of protein synthesis. As cells bear the title of living units, their mechanisms take responsibility for regulating the whole organism, and this condition is amplified by the fact that the DNA is replicated throughout all cells. Therefore, merging the knowledge of gene and protein behavior with that of genome functioning and, ultimately, with diagnose and treatment, is a highly desirable goal in the path to personalized medicine.

1.2 RESEARCH GOALS

This work tackles the subject of computational genomic evaluation and predicted consequence as a contributing step to the diagnosing pipeline. In this line, understanding the impact of genes and their properties bears great importance. In fact, it has been shown numerous times that alterations in aspects otherwise considered minimal can

introduce a great deal of change in the organism [147, 160, 165]. Therefore, we sought to understand and numerically evaluate molecular and functional characteristics to prove the hypothesis that the observed properties of a gene and other related entities can effectively be computed and used to synthetically manipulate genes and predict the phenotypical outcomes of genetic alterations.

The purpose of the research conducted in this doctorate is, therefore, to demonstrate the capabilities of gene evaluation, manipulation and learning algorithms. Thus, we focused on the creation of calculation methods to capture the properties of genetic sequences, and then on the application of those methods in the context of human variation. The overall objectives gravitate around the hypothesis being tested, as the developed methods will perform the role of observed indicators combined to assess the pathological effects of gene alterations. Thus, the main goals of the research are defined as follows.

- Create computational strategies to improve the study of genes and proteins, for instance by creating methodologies for gene and protein evaluation and manipulation that settle on top of observable biological characteristics.
- Contribute to mRNA research by tackling the challenge of gene secondary structure manipulation, which represents a major factor influencing the synthesis of proteins.
- Research machine learning methodologies to study the impact of human genetic variation.
- Collaborate with medical research groups to integrate the developed methodologies into tangible solutions for the biomedical audience.

1.3 CONTRIBUTIONS

This research contributed to the current state-of-the-art on the computer science and bioinformatics fields by advancing knowledge on the digital manipulation of genetic sequences, integrating new and known methodologies into working solutions, and by delivering new computational approaches on the growing area of human mutations. We can roughly divide the contributions into five parts (see Figure 1.3), corresponding to gene evaluation and manipulation, mRNA secondary structure studies, integration

of methods, pathogenicity prediction, and human variome exploration. I shall detail those in the following paragraphs.

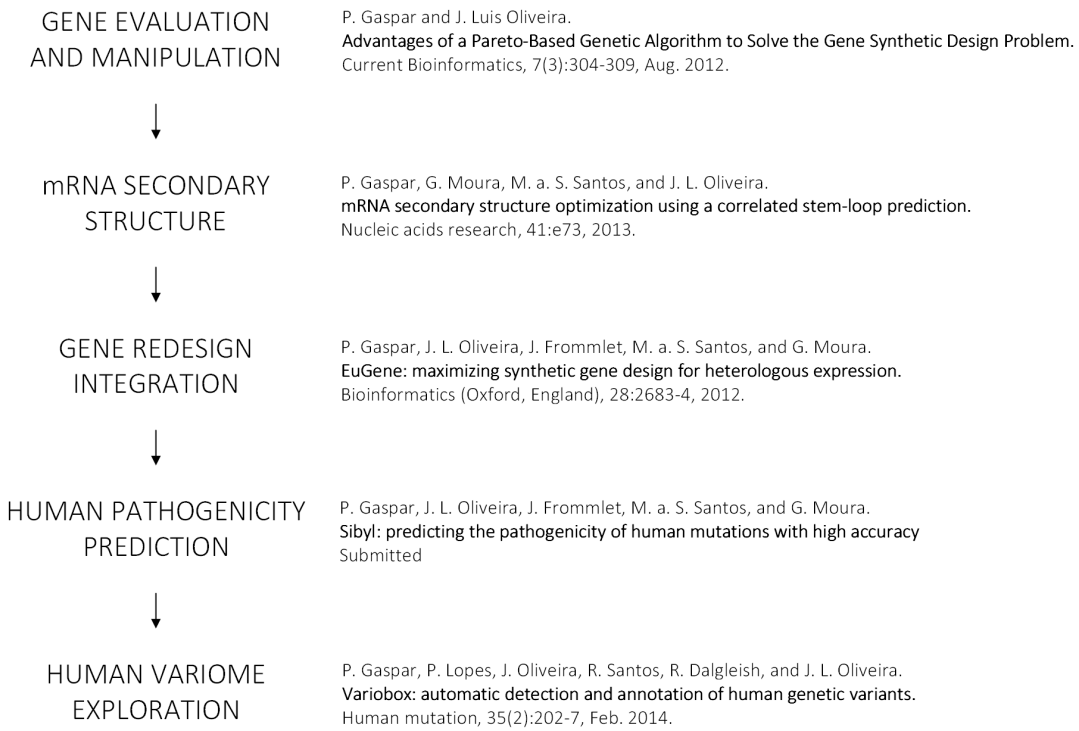


Figure 1.3 Contributions overview.

In the first contribution, we gathered from literature the main factors behind gene expression and protein biosynthesis, and developed methods to evaluate mRNA sequences on regard to those factors. These methods transform biological observations into tangible indicators, allowing, among other uses, the comparison and analysis of genes. Moreover, taking advantage of the degeneracy of the genetic code, we were able to study the combinatorial problem of modifying genes to control the indicators. Some of the outcomes of the study involve strategies developed to explore multi-dimensional solution spaces using heuristics [67, 69]. Such heuristics allow finding pseudo-optimal genes.

The second contribution comes from the field of mRNA folding. The devised research tackled the problem of secondary structures formed by nucleic acids. Given the importance and contribution of this factor to gene and protein expression, we developed a method to rapidly evaluate the minimum free energy of an RNA molecule. An algorithm was, therefore, created using heuristics to find equivalent mRNA se-

quences with minimal structural conformations [68]. By managing the optimization algorithms to control for Guanine and Cytosine content, we have also shown that codon permutations have a large impact in the secondary structure, and not only the amount of Guanine and Cytosine in the molecule. Along with the algorithm, a tool for biologists was made available to promote a facilitated use of our strategy.

The next contribution is the integration of synthetic gene evaluation, exploration and redesign functionalities into a usable solution. To ease and promote the work and access to methods and tools, we created a software system, Eugene, that also identifies genes and genomes, retrieves annotations from online repositories, performs alignments and protein secondary structure predictions. Eugene was created for biology researchers to easily analyze and redesign genes [69].

The fourth contribution lies on the subject of predicting the pathogenicity of specific human mutations. On this matter we have used machine learning in combination with the developed evaluation algorithms in order to show that mRNA characteristics have a large impact in the organism. Our solution also contributes by showing that, contrary to the widespread approach of using protein characteristics, gene characteristics can strongly contribute to prediction. As a result, we created a highly accurate system for predicting the positive/negative outcome of single nucleotide variants in human genes.

The last contribution is the integration of human gene and variome exploration capabilities, along with the pathogenicity prediction system, into a patient genetic analysis tool, Variobox [66]. To facilitate reporting new variants and findings, we also automatized the discovery and annotation of mutations, as well as the comparison of genes. This solution streamlines patient genetic examination as a step forward in personalized medicine.

1.4 THESIS ORGANIZATION

The remaining chapters of this document are organized as follows.

- **Chapter 2** will be on the background contextualization of this research, and a review on the state of the art. I will also draw a summarization of the literature on gene evaluation, and on the most recent techniques for predicting the impact of genetic alterations.
- **Chapter 3** will focus on the problem of gene evaluation by transforming litera-

ture reported characterizations into methods. The chapter will also explore the computational optimization of mRNA molecules through codon permutations using heuristics. These methods will allow quantifying observable features of genes, and redesign genes to control such features. The software package Eugene will be presented as a practical solution to integrate the developed methods.

- **Chapter 4** will expand on a particular type of gene redesign, which involves the control over the formation of secondary structures in the mRNA. This formation represents one of the most important factors guiding gene expression, and therefore I will present the solution we developed to rapidly evaluate minimum free energy, and optimize genes to maximize that energy.

- **Chapter 5** will be the culmination of the created methods, applied to a practical issue. I will present the problem of predicting the pathogenicity of the most common mutation in genes. Using machine learning, I will then show how we developed a system to tackle and solve this problem, and how we integrated it in another software package, Variobox.

- **Chapter 6** will terminate this document by presenting some concluding remarks and debate some compelling issues arising from the frontiers of this area. Future perspectives will also be discussed.

2 | FROM GENE MECHANICS TO DISEASE

In the previous chapter I mentioned the central dogma of molecular biology stated by Francis Crick. Though the term *dogma* hardly sustains itself in the scientific landscape, the theory behind his contribution still holds as the acknowledged paradigm. Nevertheless, I pointed that achievement because it bears the message of an unidirectional flow of information, intrinsic to living beings. In that sense, Crick stated that DNA generates RNA which in turn generates proteins (see Figure 2.1). And proteins are responsible for large and crucial parts of biological systems. They are simultaneously the building blocks, operators, messengers and managers of organisms.

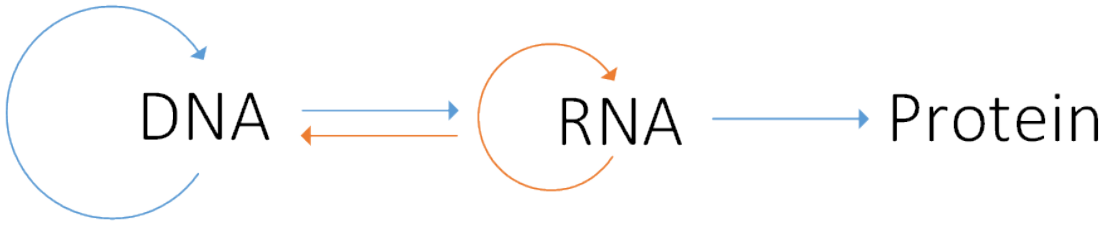


Figure 2.1 Central dogma of molecular biology. Blue arrows represent the normal transfers of information, while orange arrows are special transfers occurring only in specific conditions. DNA makes RNA which in turn creates proteins.

Naturally, when a vital manager, or any other important element, fails to perform its work correctly, the consequences might be deleterious. But according to the central dogma, this results from the effects of an upstream source of information: the RNA or DNA. It is already accepted that the RNA templates that carry the direct instructions on how to build proteins also carry indirect information that mainly serves to control the synthesis process [86, 189]. Controlling that information has become a standard way of regulating the protein creation process, and is the focus of a large share of computational biology's efforts. I will be discussing, in section 2.2, several of the

factors that influence the biosynthesis and computational strategies to evaluate and control those factors.

Another key factor resulting from the central dogma is that proteins are not directly prone to inheritance. The DNA is the single molecule that has the dual purpose of passing information to offspring and controlling the organism with that information. Therefore the functions and effects of proteins are fully described in DNA, and changes in DNA dictate the alterations at the protein level. Ultimately, analyzing the DNA of an individual is sufficient to fundamentally understand the proteome of its organism. It also means that the existence of a malfunctioning protein is coded in the DNA since the genome was initially created, and might be passed down in replication and reproduction. Of course I am largely ignoring external factors, such as environmental stress, as well as alterations occurring in cell division and other processes. However, these are largely unseen and unpredictable, and a minority compared with inherited variants. Finding and classifying inherited variants is currently the target of much research, given its potential for therapeutical and diagnosing applications [146]. In section 2.3 I discuss several of these efforts.

The next section will introduce the genetic contextualization necessary for the remaining of the document, mainly how information is carried and transformed.

2.1 GENETIC TRANSFER OF INFORMATION

The Avery-MacLeod-McCarty experiment led to the isolation of DNA (Deoxyribonucleic Acid), which is an inherited genetic library that defines most characteristics of an organism, especially physical ones [13]. The DNA molecules are present in all living beings, and are made up of four small base molecules, nucleotides (Adenine, Guanine, Cytosine and Thymine), repeated millions of times in a non-random order. In fact, the order and size of these molecules has been evolving during the past 3.55 billion years[1] as by natural selection. It is the sole arrangement of the nucleotides in DNA that dictates most of an organism's function, capabilities, physical attributes, behavior, and even cognitive skills.

Evolution led to the development of small clusters of nucleotides combined in a specific order, with predefined tasks, called genes. Since the DNA only carries information, the transformation of genes into useful functional blocks is performed by

[1] Throughout the document I refer to the short-scale billion, i.e. 10^9

other entities. In the nucleus of the cell, where the DNA is stored in eukaryotes, genes are copied from the DNA by an enzyme called RNA polymerase, in a process called transcription (Figure 2.2a). This is the first step of a larger process called gene expression, whereby the information carried by DNA is interpreted. The result of this step is another nucleic acid molecule released into the cell nucleus, the precursor mRNA, an exact copy of the gene except all Thymine nucleotides are replaced by Uracil nucleotides. The precursor mRNA then undergoes a series of preprocessing steps such as modifying its ends to avoid degradation, and most importantly, performing splicing. Not all of the precursor mRNA is necessary; in fact, the majority of a gene is made of large sequences of nucleotides called introns that are removed in the splicing step (Figure 2.2b). After that only the information-bearing regions, exons, are kept and joint together to assume the final form, the mature mRNA (messenger RNA).

Note that the mRNA is the central molecule of this research, as much of the computational analysis and manipulation is made on digital representations of it, often in the form of strings of A, C, U and G, representing each type of nucleotide.

The final step of gene expression is translation, and it is a very complex process involving many entities in the cell (Figure 2.2c). The goal is to build a protein with the instructions coded in the mRNA. A complex called ribosome attaches to the mRNA and reads groups of three consecutive nucleotides at a time, called codons, starting in a specific codon named the start codon. The ribosome then reads the entire coding region of the mRNA, until it reaches another special codon, the stop codon, where it ends the translation and detaches from the mRNA. Each codon read by the ribosome is decoded into an amino acid, and the amino acids are added together to produce a polypeptide chain, which is the final product that the gene coded for.

Naturally, since a codon is a group of three consecutive nucleotides, the complete coding sequence has a number of nucleotides that is multiple of three. Also, the same codon is always decoded to the same amino acid, and since there are four different nucleotides the number of different combinations of codons is $4^3 = 64$. However there are only 20 distinct amino acids, and so several different codons can decode to the same amino acid. As a result of this redundancy, there are on average 3 codons capable of decoding each amino acid. Codons translating into the same amino acid are called synonymous codons. This is called the degeneracy of the genetic code, and is a fundamental property that is much explored in this work. It should also be noted that stop codons do not decode any amino acid, but there are generally three

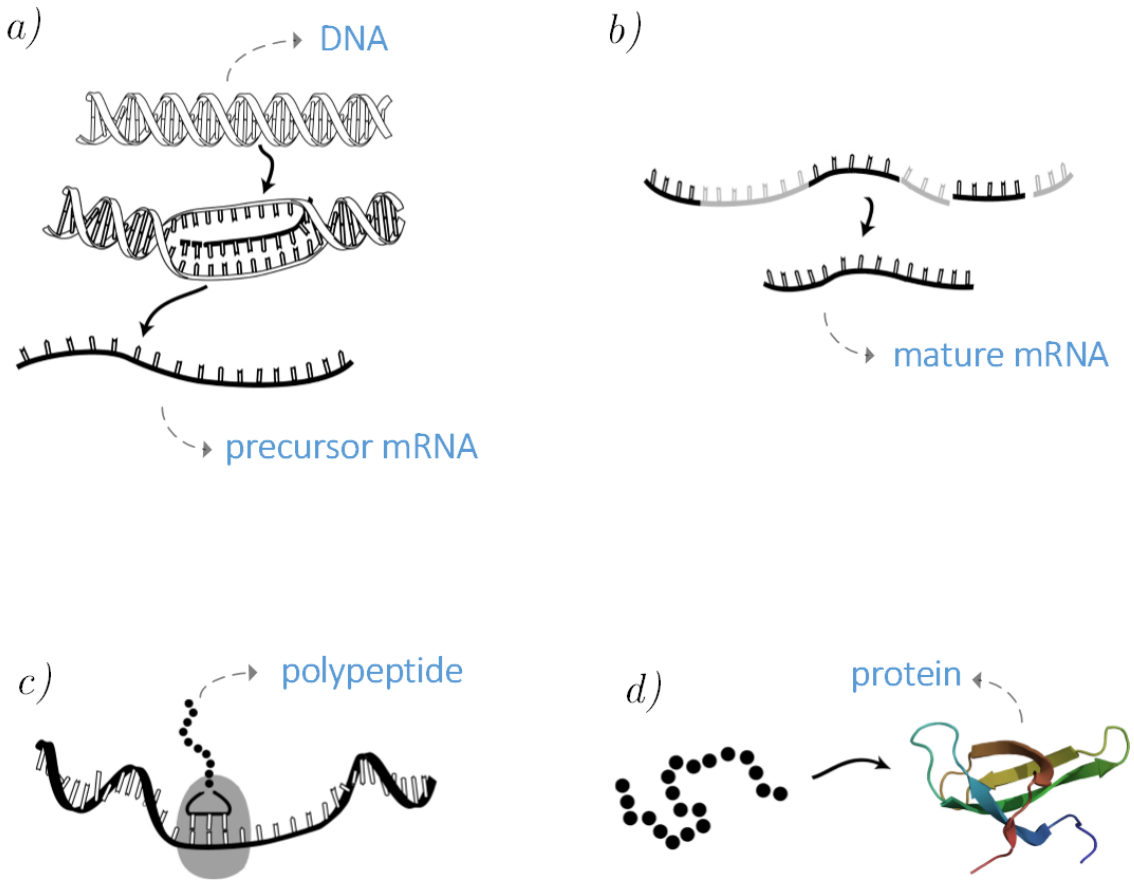


Figure 2.2 Diagram of the steps of gene expression. In **(a)** the DNA is untwisted and then a gene is copied by an external complex called RNA polymerase. The result is the precursor mRNA. Next, in **(b)**, splicing occurs removing intronic regions from the pre-mRNA and stitching together the remaining exons, making the final version of the messenger RNA. In **(c)** the ribosome (in gray) reads the mRNA, codon by codon. For each codon, an external molecule, the transfer RNA (tRNA), brings the correct amino acid to the ribosome, attaching itself to the codon, leaving the amino acid and then detaching. The translation occurs by linking one amino acid at a time, forming a chain called polypeptide. As the polypeptide leaves the ribosome and interacts with the cytoplasm (the liquid interior of the cell) it folds to assume its three dimensional conformation. That step is represented in **(d)** where, even after the translation is over, the folding proceeds with the help of other cell mechanisms.

variants of stop codons (this can change from species to species). Moreover, changes in genes can be made by replacing codons with synonymous ones, and this is the main mechanism behind the redesign of genes. Though the genetic code is almost the same in all species, there are slight variations in what codons decode for which amino acids. Table 2.1 shows the standard genetic code.

Amino acid	Codons
Ala/A	GCT, GCC, GCA, GCG
Arg/R	CGT, CGC, CGA, CGG, AGA, AGG
Asn/N	AAT, AAC
Asp/D	GAT, GAC
Cys/C	TGT, TGC
Gln/Q	CAA, CAG
Glu/E	GAA, GAG
Gly/G	GGT, GGC, GGA, GGG
His/H	CAT, CAC
Ile/I	ATT, ATC, ATA
START	ATG

Amino acid	Codons
Leu/L	TTA, TTG, CTT, CTC, CTA, CTG
Lys/K	AAA, AAG
Met/M	ATG
Phe/F	TTT, TTC
Pro/P	CCT, CCC, CCA, CCG
Ser/S	TCT, TCC, TCA, TCG, AGT, AGC
Thr/T	ACT, ACC, ACA, ACG
Trp/W	TGG
Tyr/Y	TAT, TAC
Val/V	GTT, GTC, GTA, GTG
STOP	TAA, TGA, TAG

Table 2.1 Standard genetic code table. Each amino acid can be decoded by several codons, but each codon only decodes into one amino acid. Note that the start codon also codes for the Methionine amino acid, and that stop codons do not decode into any amino acid (they only signal the termination of translation).

When the translation process is over, the protein (still just a polypeptide chain) is not yet fully capable. As the chain leaves the ribosome, it starts folding on itself, driven by the hydrophobicity and chemical interactions of its amino acids, until it achieves its natural conformation, with minimal free energy [7] (Figure 2.2d). The folding process is complex and is the target of a large amount of current research in the field of computational biology. The acquired shape is essential to ensure the functionality of the protein, as it allows exposing the correct interfaces to interact with other elements of the organism. Moreover, incorrect folding may cause the protein to lose some or all of its function, perhaps even rendering it toxic. For instance, neurodegenerative diseases such as Alzheimer's and Parkinson's are believed to be caused by the accumulation and aggregation of incorrectly folded proteins in neurons [164]. Incorrect folding can easily occur when there is an alteration in one or more of the amino acids of the protein, if that alteration generates significant physicochemical property changes. Changes in amino acids are generally caused by mutations in the

genes, which in turn can be caused by numerous factors, such as replication and reproduction errors. It should be noted that alterations in genes (and proteins) are not mandatorily deleterious, and are indeed one of the mechanisms for evolution.

2.2 SYNTHETIC GENE EVALUATION AND REDESIGN

The evolutionary constraints that shape the genome of an organism have long been the target of a large portion of the scientific community in biology [79, 133, 166]. These constraints are intrinsically related with the mechanisms of gene expression and, ultimately, with the function of proteins. Understanding their impact not only expands the knowledge regarding the mechanisms of genomes, but also paves the way for synthetic gene manipulation.

As mentioned in the previous section, the sequence of nucleotides in the mRNA defines the output of the translation: the chain of amino acids. I shall argue that this sequence serves several simultaneous purposes besides that of translating, and that such secondary functions can be computationally measured and used to algorithmically optimize genes. In fact, Post et al. in 1979 were among of the first to report a parallel set of information in the coding sequences of mRNAs, not directly related with the translation [148]. They noticed that there was a strong correlation between the usage of codons and the amount of cognate tRNAs. That bears some explanation: codons, the decoding unit which is made of three nucleotides, exist in different amounts throughout all the genes of the DNA of an organism; the probability of finding each codon is not random, nor it is equal among all codons, meaning some codons are more common than others (see Figure 2.3) [84]; tRNAs are the cell entities responsible for bringing amino acids to the ribosome to create the polypeptide chain; there are different tRNAs for different amino acids, and having more tRNAs of one type makes the codons for that amino acid decode faster, because the chances of having a tRNA in the surroundings of the ribosome are higher; Post et al. found that the frequency of each codon is highly correlated with the amount of each type of tRNA, and therefore with the speed of translation.

2.2.1 GENE EXPRESSION AND QUALITY INDICATORS

Codon usage was one of the first characteristics to be widely calculated on a gene, besides those of function assessments. It allows estimating how fast a gene will be

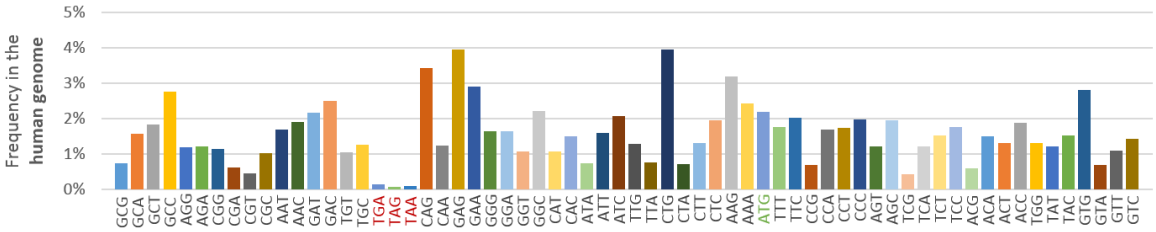


Figure 2.3 Codon usage distribution in *Homo sapiens*. The bars represent the 64 possible codons, with a height corresponding to their frequency in the human genome. Some codons are extremely uncommon, such as the stop codons (red), which are used only once in every gene. A biased landscape is clearly visible.

decoded into a polypeptide by analyzing the frequency with which each codon of the mRNA is used in the whole genome. Genes with common codons will have more tRNAs available at the time of translation and thereby will create proteins faster [37, 195].

This evaluation gave room for further understanding the reasons underlying the biased evolutionary choice of codons in genes. For instance, in 2009, Zhang et al. reported that the translation speed signatures, that is, the different zones of the gene that decode slower or faster, can affect the efficiency of the folding step [201]. They showed that regions of the gene with less frequent codons[2] temporarily arrested the ribosome during translation, waiting for tRNAs, and that slower rate allowed the already created part of the polypeptide to fold correctly. Conversely, regions with frequent codons were decoded faster. This means that the frequency of each codon has a role that indirectly guides the creation of the protein [140]. Zhang et al. also showed that replacing low-frequency codons by ones with a higher frequency can negatively impact protein expression, by removing necessary pauses in the translation. For instance, a transparent case of the role of codon usage is the difference between γ-actin and β-actin proteins: both proteins are 98% similar in their amino acid sequence, but have distinct functions within the cell; it has been shown that their difference is due to codon usage differences that control the rhythm of translation [197].

Another reason for the existence of a codon bias is gene expression itself. Genes vary widely in their expression, that is, in the amount of protein produced. This variability is related with the necessity to produce proteins in different quantities due to their function in the organism. Thus, it has been shown that codon usage is used to regulate expression levels, where genes with frequent codons being translated often

[2] Actually less abundant tRNAs, which is correlated with codon frequency [83, 101].

and at high volumes [72, 85, 156]. The large impact of codon usage has made it a preferred strategy to evaluate genes, and the base methodology for many other forms of evaluation and redesign.

For instance, Eyre-Walker and Bulmer [55], as well as other authors [183], noticed that the beginning of the gene plays an essential role in providing the correct circumstances for the translation to occur correctly. They observed that the initial region of translation in genes is mainly populated by low-frequency codons, purposely slowing decoding initially, in a *ramp effect*. More recently, it has been shown that this effect is also related with mRNA structure [20].

Another example is the *codon correction effect*, by which codons used at the beginning of a gene to encode amino acids, will be preferentially reused along the gene, or at least the same tRNAs [31]. As mentioned before, each amino acid can be encoded by several different codons, called synonymous among themselves. Cannarrozzi et al. observed that a single gene tends on using the same synonyms for each amino acid, with the reasoning that this is energetically favorable and less prone to errors, as the tRNAs are already nearby the ribosome and therefore the translation will be faster.

Though codon usage was shown to be one of the most important factors involved in gene decoding, the developments made in the field have shown that many gene characteristics influence the production and functionality of proteins. Another such factor is *codon context*, which is to some extent similar to codon usage, but relates to pairs of codons instead. It states that the choice of a codon is modulated by the surrounding nucleotides and codons [48, 125]. More specifically, codon context is the nonrandom utilization of adjacent codon pairs [179, 200] which is related to steric interactions between consecutive tRNAs in the ribosomes [74, 171]. Ultimately, as with codon usage, codon context is correlated with the translation rate and accuracy [42, 124].

The amount of guanine and cytosine (*GC content*) in the genome and genes has also been shown to be related with the expression of genes [113, 152]. It was suggested that this relation may be due to the higher stability of guanine-cytosine molecular pairs, favoring a higher accuracy during translation, and even driving codon usage bias [96]. Cytosine-Guanine sites in genes can also be methylated, a chemical mechanism that can turn a gene off, and has been shown to be related with the expression of DNA vaccines, for instance, in HIV treatment [99]. Moreover, given the stability of the GC pairs, genes with larger amounts of GC content are more prone to the formation of secondary structures in the mRNA.

Secondary structures are another important factor contributing to gene expression. In DNA, Adenine nucleotides pair with Thymine, and Guanine nucleotides with Cytosine. In RNA, the effect is the same[3], but since RNA molecules are much smaller and single stranded, they eventually end up folding on themselves. The consequence of this phenomenon is the possibility of forming large stretches of paired nucleotides, called secondary structure (the primary structure being the sequence of nucleotides itself) [76]. In some occasions, this is a normal and desired behavior, such as with tRNAs, where the formation of their secondary structure is what grants them their functionality. However, in normal protein-coding genes, the formation of secondary structures can hamper the work of the ribosome, causing pauses along translation or even preventing the initiation of translation [20, 50, 104, 202]. This effect has a large impact and is thoroughly explored in chapter 4.

Furthermore, errors during gene expression may be deleterious for the functional maintenance of cells. A source for errors during the decoding step is the loss of the reading frame (frame-shifting), whereby the ribosome shifts the reading of codons by one or two nucleotides and erroneously proceeds the decoding until another stop codon is found. One factor causing the slippage is due to the presence of long chains of repeated nucleotides or repeated patterns of nucleotides [27]. However, the gene has a mechanism to avoid the wasted consumption of cell resources (e.g. amino acids) when frame shifting or other similar translation errors occur. As mentioned before, a subset of codons is used in the genome to indicate the end of the translation region: the stop codons. It has been shown that genes have evolved to include an overwhelming amount of stop codons at out-of-frame positions, as this strategy prematurely stops incorrect translations when frame-shifting occurs [144, 163]. An example of this protection system is observed in the genome of *Escherichia coli*, where in nearly half of the genes the start codon (ATG) is followed by a codon starting in Adenine, forming the tetranucleotide ATGA. If frame-shifting occurs initially, the translation is halted immediately because a *hidden stop codon* is found (TGA), preventing cell resources from being used in an incorrect translation.

Another factor involved in controlling protein synthesis, and eventually even causing translation errors, is the interaction of the mRNA with other cell machinery. For instance, in *Escherichia coli*, a toxin called MazF regulates growth by cutting mRNA molecules at ACA sites [15, 203]. Another example is Shine-Dargarno or Kozak sequences, which are specific nucleotide sequences present in the mRNA before the start

[3]Though in RNA Thymine is replaced by Uracil, which still bonds with Adenine.

codon, allowing the ribosome to initially attach itself to the mRNA [103, 170]; when these sequences are found in the coding region, ribosomes might attach to the mRNA, hampering any ongoing translation [87, 172]. Also, as seen in Figure 2.2b, a splicing step splits the pre-mRNA to remove unnecessary regions. Splicing regulatory regions are sequences within the gene that silence or enable splicing, and therefore have a large impact in expression [189]. This has been shown to be associated with a large portion of inherited diseases [123]. The presence of these and other sequences that enable the interaction with cell machinery can be largely deleterious to the expression of a gene.

The factors presented in this section are perhaps the ones that mostly influence gene expression, and it is important to note that they can be directly measured from the mRNA. However, the mechanisms that are involved and regulate protein synthesis are not isolated from the remaining processes of the cell. Therefore there is a substantial amount of still unknown, highly variable or hardly mensurable factors that also impact the creation of proteins.

2.2.2 ANALYTICAL EVALUATION METHODS

In spite of the knowledge that gene expression indicators are an important source for analysis, methods for the evaluation and manipulation of genes are still in an embrionary phase: since genes are central to living systems, the creation or redesign of genes and proteins are intricate procedures involving the whole complexity of organisms.

Nonetheless, with the sequencing of genomes becoming streamlined, the availability of DNA in digital format, from many species, has allowed to perform comprehensive statistical analysis both within species and among species, unveiling patterns of evolution and functionality. Taking advantage of the degeneracy of the genetic code, several strategies to evaluate and redesign genes have emerged, based on the previously described gene expression and quality indicators.

To measure codon usage, the most simple method is measuring the frequency of each codon in the whole genome. However, Sharp et al. proposed a different method that takes into consideration the frequencies of the amino acids, since the amino acid distribution is not random [167]. Their method, the relative synonymous codon usage (RSCU), measures the deviation of the observed frequency of a codon from the expected frequency if codon usage was uniform among synonymous codons (equation 1).

$$RSCU_{i,j} = \frac{F_{i,j}}{\frac{1}{n_i}\sum_{k=1}^{n_j} F_{k,j}}$$

Equation 1 The RSCU for codon j, is based on the frequency $F_{i,j}$ of codon i for amino acid j, and on the frequency of every other codon that decodes to the same amino acid.

Though the RSCU is an improvement over the simple frequency because it takes into account the usage of synonymous codons, Sharp and Li have noted that this and other similar measures (e.g. codon preference statistic, codon bias index) lack the ability to be comparable among different genes, due to the difference in amino acid composition [166]. Thus, they proposed a method, Codon Adaptation Index (CAI), based on the fact that some genes are highly expressed (often called house-keeping genes), and that evolution has strong selective constraints in these genes that favour translational efficiency. Such pressure has led the genes to the use of highly efficient codons. CAI uses the codon usage of those genes to calculate a score of bias, measuring the relative distance of any gene to the house-keeping genes (equation 2).

$$\begin{aligned} w_{i,j} &= \frac{RSCU_{i,j}}{RSCU_{imax}} \\ CAI &= \left(\prod_{k=1}^{N} w_k\right)^{\frac{1}{N}} \end{aligned}$$

Equation 2 The CAI of a gene is formulated from the geometric mean of the relative adaptiveness $w_{i,j}$ (for codon j and amino acid i) of each of its N codons/amino acids. Note that CAI measures a whole gene, rather than a single codon.

Some authors have created modified versions of this approach, for instance the tRNA Adaptation Index (tAI), which is based on the fact that codon usage is highly correlated with tRNA abundance, which in turn is highly correlated with the number of genes in a genome that encode each tRNA [53]. The relative adaptiveness w of tAI modifies the original one as shown in equation 3.

$$w_i = \sum_{j=1}^{n_i} (1 - s_{i,j}) tGCN_{i,j}$$

Equation 3 The tAI of a gene is formulated from CAI, changing only the relative adaptiveness $w_{i,j}$. n_i is the number of tRNAs for codon i; $tGCN_{i,j}$ is the number of genes encoding the jth tRNA for the ith codon; and $s_{i,j}$ represents the efficiency of the tRNA in attaching to the codon.

However, both CAI and tAI are species-dependent measures, given that the set

of optimal codons differs from species to species. To tackle that problem, Wright proposed a new method, the Effective Number of Codons ($\hat{\text{Nc}}$, shown in equation 4), which measures the extent to which the codons used in a gene are biased [199]. This measure varies between 20, when a single codon is used for each aminoacid, to 61, when synonymous codons are equally used.

$$F = \frac{(n\sum_{i-1}^{k} p_i^2)-1}{n-1}$$

$$Nc = \sum_{i=1}^{6} \frac{i}{F_i}$$

Equation 4 The effective number of codons is calculated by first quantifying codon homozygosity F for each amino acid. F_i is the average of codon homozygosity for all amino acids with i synonymous codons.

Furthermore, codon context, as discussed previously, also has a large impact in the translation efficiency. Changing the codon context has been shown to be an effective strategy to attenuate the action of virus. Coleman et al. have proposed a method, the Codon Pair Bias (CPB), to measure the bias of codon pairs beyond simple frequency by determining if the codon pairs of a gene are on average under or over-represented in a genome (equation 5) [41].

$$CPS_{a,b} = ln\left(\frac{F_{a,b}}{\frac{F_a F_b}{F_x F_y} F_{x,y}}\right)$$

$$CPB = \sum_{i=1}^{k} \frac{CPS_i}{k-1}$$

Equation 5 Codon Pair Bias is the average of codon pair score for every pair of codons i in a gene. The codon pair bias (CPS) measures the observed frequency F of a pair of codons a, b compared to the expected when taking into account the frequency of the pair of their corresponding amino acids x, y.

Codon context is independent of codon usage in the sense that even codon pairs made of codons with large values of codon usage can be very infrequent. Other methods, such as Pearson's chi-squared tests, have also been used to measure the significance of codon pairs bias [39], though CPB has been widely adopted. In the research of Coleman et al., the codon context was changed in the gene encoding a specific protein of a virus. The change into under-represented pairs of codons led to a decreased rate of translation, and therefore attenuating the virus.

Assuming that no errors occur in the translation process, codons in genes can be safely replaced by synonymous ones guaranteed that the resulting polypeptide will

remain the same. However, as discussed in section 2.2.1, other factors contribute to the correct formation of the final protein. Nonetheless, the base for redesigning a gene while maintaining the amino acid sequence is codon substitution, and Coleman et al. applied that method by searching for the best combination of synonymous codons for each position of the gene that decreased the value of CPB. Analogously, optimizing the codon usage of a gene can be achieved by replacing every codon of the mRNA by a synonymous with a larger value of CAI, RSCU or other measure of codon usage.

Being able to artificially create or redesign genes that are able to produce proteins faster, and consequently in larger quantities, is a goal of research and industry [196]. Besides academic applications such as the need for large amounts of diverse proteins, large scale production of proteins is a necessary step for therapeutic ends such as vaccine creation and drug design [64]. For instance, the RSCU was recently used with success by Hashaam et al. to redesign the Interferon-λ gene in order to express it in larger quantities to be used as a treatment for Hepatitis C [4]. Moreover, the creation of proteins in large quantities is usually performed resorting to host species such as the bacteria *Escherichia coli*, which allow fast growth rates, inexpensive media and well understood genetics [29]. This process is called heterologous expression, and is performed by placing the target gene in host species, so it recognizes the gene and translates it to proteins as if it were from its own DNA. One vulgar example of this process is the mass production of insulin, the protein responsible in humans for regulating carbohydrates in the blood.

However, in spite of the advantages of using a different host to perform translation rapidly and in large amounts, there are several obstacles in expressing a gene (e.g. a human gene) in a strange host species. Firstly, the genetic code and cell machinery might be different, and thus some codons might translate to different amino acids. As a result, the gene would produce a different protein, likely non-functional. Another obstacle is the fact that codon usage varies widely among species, and therefore a codon that is frequent and yields a rapid translation in the source organism might even be very infrequent in the host species, making that codon slow-decoding. As discussed in section 2.2.1, the codon usage signature is an important factor in translation, controlling the velocity of the protein synthesis. In fact, Angov et al. showed that adapting the target gene's codon usage to the host species could yield a ten-fold increase in the amount of protein expressed, compared to using the native sequence [8, 9]. This was performed by an algorithm named codon harmonization. In codon harmonization, the codons of the native gene are replaced by codons that in the host

species decode into the same amino acid and have similar codon usage. This way, the decoding speed signature remains the same as in the original organism, granting a correct folding of the resulting protein. In Angov et al. study, a protein to potentially fight malaria which otherwise had no expression in *E.coli* was successfully expressed after codon harmonization. Equation 6 determines the extent to which a foreign gene is adapted to the host organism, largely based on the methodology applied by Angov et al.

$$\text{Harm} = \min \sum_{i}^{N} |F_{native}(i) - F_{host}(i)|$$

Equation 6 Evaluation of codon harmonization for a gene with N codons. By performing codon substitution, the algorithm uses this function to reflect how the codon usage of each codon is deviated from the native codon usage. F can be any measure of codon usage, such as RSCU.

As discussed previously, another factor that largely influences gene expression if the formation of secondary structures in the mRNA. The secondary structure of molecules is commonly measured by the minimum free energy resulting from the most stable structure. The most stable structure is acquired when the maximum number of nucleotides is paired. Numerous algorithms and approaches to predict secondary structures have been created [98, 111, 176, 205], but the most sophisticated and well known is the fast dynamic programming approach from Zuker and Stiegler [206], which is based on a first approach from Nussinov et al. [134], and served as the basis for more recent methods. While Nussinov et al. approach was simply intended to maximize the number of base pairs, Zuker and Stiegler created a more realistic algorithm by creating a model of Gibbs free energy of the RNA molecule and then trying to minimize that energy. This algorithm has a time complexity of $O(n^4)$, though it can be reduced to $O(n^3)$ with some optimizations, and a memory complexity of $O(n^2)$. Implementations can be found in the mFold [205] or Vienna RNA [80] software packages. It is important to note that though these algorithm try to maximize the number of base pairs formed in the secondary structure of the RNA, the original sequence is never altered, only the potential folding is.

2.2.3 COMPUTATIONAL REDESIGN

In section 2.2.2, codon usage harmonization was shown to optimize genes for heterologous expression. However, harmonization is not the only method for redesigning

genes. Indeed all other gene indicators can be optimized by performing codon substitutions, as long as there is a form of evaluating the indicators. For simple indicators, such as codon usage or GC content, gene optimization can be achieved by successively altering codons by synonymous that increase their evaluations, e.g. RSCU or the number of Guanine and Cytosine nucleotides. However, for more complex indicators that cannot be reduced to single codon evaluations, such as the mRNA secondary structure which results from the whole gene, using codon-by-codon substitutions is not a feasible strategy: altering a codon influences the whole structure. Furthermore, exploring the complete set of codon combinations to look for the best fit is computationally impossible as there are on average 3^n possible permutations for a gene with n codons. A peptide with ten amino acids can be coded by an average of 59 thousand possible synonymous codon sequences. The problem expands when there are several simultaneous constraints, such as improving codon usage while avoiding nucleotide repeats that cause ribosomal slippage. Several solutions have been presented to overpass this issue, such as randomly generating many synonymous sequences and then choosing one with optimal indicators, or using heuristics to search faster through possible combinations.

Search algorithms have already been shown as an effective solution for the computational redesign of genetic sequences [157, 188]. Indeed the latest method employed by Angov et al. to achieve codon harmonization was to perform a stochastic search using an algorithm called simulated annealing to explore possible synonymous gene configurations [9]. Moreover, Chung and Lee have also recently used genetic algorithms to simultaneously optimize a gene for codon context and codon usage [39]. Using search algorithms for gene optimization is a topic thoroughly explored in chapter 3.

Nonetheless, some available software packages already allow optimizing genes for several expression indicators. For instance Gene Composer, a very complete protein constructor, allows the back-translation of an amino acid sequence into a codon sequence [151]. It allows defining the values for specific indicators, to be used in the back-translation, such as the percentage of GC content. However, the back-translation and optimization processes are achieved by generating thousands of synonymous sequences, and selecting the one that most closely matches the target GC% [112]. Repetition removal, inserting out-of-frame stop codons, removing deleterious sites and other indicators are also available in Gene Composer in a stepwise manner achieved by targeted synonymous codon replacement. Other applications, such as Gene De-

signer from DNA2.0 and IBG GeneDesigner, use genetic algorithms to back-translate a sequence according to a set of constraint parameters [187, 188]. GASCO, a software dedicated to gene expression optimization, also uses genetic algorithms to enhance codon usage as well as including and avoiding sequence motifs [157]. Using heuristics grants these applications the advantage of a more thorough search for candidate solutions. Optimizer is another software, available through a web page, focused on the redesign of genes for codon usage enhancement [150]. It uses several techniques, such as a Monte-Carlo based heuristic, and a method designed to achieve maximum codon usage with minimum changes to the original sequence. It also offers codon usage tables for the highly expression genes of many species, which allows a researcher to perform optimizations to adapt the codon usage to the specific machinery of a species. Optimization, however, is limited to codon usage, and using personalized or different genomes than those supplied is out of the scope of the software. Similar features are found in Codon optimizer, which also provides calculations for CAI, and identifies regions of interaction with cell machinery (e.g. restriction sites) [63]. Other applications, such as GeneDesign [153], DNAWorks [81], and JCat [73], offer tools to interact and design genes in a step-wise manner, though the integration of expression indicators is limited.

An overview of the main available software tools for gene optimization and their inclusion of gene expression indicators is depicted in Table 2.2.

	Codon Usage	Harmonization	GC Content	mRNA Secondary Structure	Site avoid/include	Hidden stop codons	Nucleotide repeats	Heuristics
GASCO	Yes	No	Yes	Unknown	Yes	No	No	Yes
Optimizer	Yes	No	Yes	No	Yes	No	No	No
JCat	Yes	No	No	No	Yes	No	No	No
Codon optimizer	Yes	No	No	No	Yes	No	No	No
Gene Composer	Yes	No	Yes	Limited	Yes	Yes	Yes	No
Gene Designer	Yes	No	Yes	Limited	Yes	No	Yes	No
IBG GeneDesigner	Yes	Unknown	Unknown	Unknown	Yes	Unknown	Unknown	Yes
GeneDesign	Limited	No	No	No	Yes	No	No	No
DNAWorks	Yes	No	No	Limited	Yes	No	No	Yes

Table 2.2 Gene optimization software comparison. All the tools offer a form of codon usage, though GeneDesign only offers a frequency-only based approach. Harmonization is not present in any of the applications. Some limited approaches to mRNA secondary structure optimization exist, though poorly based in stepwise algorithms, or relying in external tools (e.g. mFold). Only Gene Composer offers an algorithm to insert out-of-frame stop codons.

Though all tools employ more than one gene expression indicator to redesign codon sequences, most have only a very limited set of methods. Furthermore, only GASCO,

IBG GeneDesigner and DNAWorks use heuristics to perform the optimization. The remaining applications resort to iterative approaches where a gene is changed step-by-step to control one indicator at a time, while trying to maintain the results achieved in previous steps for other indicators.

2.3 MENDELIAN PATHOGENICITY

One of the main mechanisms for evolution is genetic mutations. Ironically, this mechanism is also the main cause of disease. While genes are meant to encode the sequential formulas to build proteins and replicate themselves to offspring, they are often[4] subject to alterations. The alterations can occur under several circumstances, such as replication errors when the DNA is copied, or environmental influence such as UV rays, chemicals and radiation. In humans, however, only alterations originating in specific cells (germ cells) cause the alteration to be passed to offspring. Once an alteration is passed, there is a risk that it becomes part of the DNA of the following generation and it can be inherited from that point on into subsequent generations. Humans have two copies for each chromosome, one from each parent. This means that a hypothetical genetic alteration may exist only in one copy of the chromosome and not the other when inheriting the alteration from only one of the parents. This alteration might be pathogenic, i.e. disease-causing. However, having two copies of each chromosome grants humans with the ability to still express the gene correctly from the non-pathogenic allele (one of the copies of the gene). Nonetheless, many diseases can still express themselves with a single copy of the alteration. These are called autosomal dominant diseases, and examples of these are Huntington's disease, osteogenesis imperfecta (also known as brittle bone disease), and neurofibromatosis. On the other hand, when two copies are needed for the disease to happen, the disorder is called autosomal recessive, and examples of these include cystic fibrosis, and sickle-cell anaemia.

Until now I have only mentioned monogenic traits (i.e. that occur by the alteration in a single gene, also known as Mendelian), but diseases can be extremely complex and depend on multiple alterations in several genes or regions of the DNA. For instance, rheumatoid arthritis, which is the most common rheumatic disease affecting approximately 1% of the population, has its etiology linked with two or three genes

[4]The rate of human mutations has been shown to be broadly $\sim 2.5 \times 10^{-8}$ per nucleotide per generation [126].

as well as environment and lifestyle factors [145]. Diabetes, Alzheimer's and cancers are also examples of polygenic disorders, and there are, generally, more polygenic disorders than monogenic [38]. Moreover, in the Online Mendelian Inheritance in Man (OMIM) database there are 5,202 monogenic phenotypes for which the molecular basis is known, and 3,195 human genes with known phenotype-causing mutations [77]. OMIM also reports 1,707 described phenotypes for which the molecular basis is unknown, and 1,856 phenotypes with suspected mendelian basis. It is fair to assume that the etiology of phenotypes, including diseases, is an active field still giving its first steps. It is also worth mentioning that studies of monogenic diseases contribute a great deal to the knowledge of polygenic diseases through the understanding of the mechanisms of pathogenicity and gene regulation [10].

2.3.1 MECHANICS AND CONSEQUENCE OF MUTATIONS

Genetic mutations (also known as variants or alterations) can occur in distinct ways, varying in size and form. The most simple and common variation is the single nucleotide variant (SNV), whereby a nucleotide in the DNA is replaced by another. When an SNV occurs within a coding region of the DNA it will affect a codon and therefore it may alter the translated amino acid if the mutated codon is not a synonymous of the original codon. These variants are called non-synonymous SNVs (nsSNV), and are opposed to synonymous SNVs (sSNV). Because nsSNVs directly affect the structure of the protein, they are believed to have a larger impact on human health compared with sSNVs [130]. Other types of variants include:

- **Insertion** When one or more nucleotides are inserted in the DNA between two existing nucleotides. The impact of an insertion, even a single nucleotide, in a coding region can be very deleterious to the expression of the gene, as a frame-shift occurs and following codons are read incorrectly.

  ATG **GTC** GGT **CAA**... ⟶ ATG **AGT** CGG **TCA** A...

- **Deletion** When one or more nucleotides are removed for the DNA. The consequences can be similar to those of an insertion when occurring withing a coding region.

  ATG **GTC** GGT **CAA**... ⟶ ATG **GTC** GTC **AA**...

- **Deletion-Insertion** As the name indicated is the combination of the previous two mutations types: a region of the DNA is replaced by one or more nucleotides.

ATG **GTC** GGT **CAA**... ⟶ ATG **GTC** GAC **GGG** A...

- **Inversion** Occurs when a nucleotide sequence in the DNA becomes reversed.

  ATG **GTC** GGT **CAA**... ⟶ ATG **GTC** CTG **GAA**...

- **Duplication** When a nucleotide, or a sequence of nucleotides, is copied one or several times.

  ATG **GTC** GGT **CAA**... ⟶ ATG **GTC** GGG **GGT** CAA...

Though the DNA of any individual is 99.5% similar to every other, we look, paradoxically, very different [107]. This happens because some small changes have large phenotypic effects. At the heart of these changes is the effect that a mutation can cause to the expression of a gene[5]. As discussed previously, a variant affecting a codon can alter the primary structure of the resulting protein (i.e. the amino acid sequence). The consequence of the alteration varies depending of many factors, such as differences between the original and the new amino acid and, as shown in section 2.2.1, differences between the original and new codons. It is the order of the amino acids that mainly dictates the final structural conformation that the protein will have, and it is the structure that defines its functionality and role. Amino acids have different physicochemical properties, such as mass, volume, charge, hydrophobicity (the ability to repel water), and polarity. These properties, mainly hydrophobicity and the interaction between amino acids, guide the folding process by obligating water-repelling amino acids to enclose themselves in the nucleus of a protein, leaving hydrophilic amino acids in the exterior (also performing function), and at the same time minimizing the free energy of the molecule conferring stability to the structure [16, 52, 138]. By replacing an amino acid, the properties of the new amino acid might result in a disruption of the original folding, or disabling of critical interaction sites, rendering the protein dysfunctional. Besides the fact that the function originally intended to be performed by such a malformed protein will no longer be accomplished (or will be in a limited fashion) and therefore having a chance of being pathogenic, it has also been shown that misfolded proteins can aggregate, forming fibers, and initiate profound cellular dysfunction [26, 155]. This is the basis for diseases such as Parkinson's, Huntington's and Alzheimer's [164].

[5]It should be noted that though I focus in changes occurring in coding regions to explain phenotypes, there is large evidence that changes in non-coding regions are also responsible for regulating the expression of genes.

As mentioned previously, differences are not limited to gene products. In fact, the majority of the expression indicators discussed in section 2.2.1 can be involved in determining the consequences of a genetic mutation. For instance, a variant altering a codon into a synonymous one with very different codon usage might provoke a deep change in the translation speed, affecting the ongoing protein folding process that occurs while translation happens [9, 100, 201]. Furthermore, variants occurring in the initial coding region of the mRNA might affect the initiation of translation (see *ramp effect* in section 2.2.1). Also, variants affecting splicing regulatory sites have been shown to influence the pre-mRNA splicing step, with studies reporting that 16% of disease causing mutations affect these sites [123, 189].

It should be noted that it is hard to deduce any determinism from the mechanisms of mutations and gene expression, as they are complex processes depending on a large myriad of variables, spanning from genetic to environmental factors. Variables such as demographics, role of the gene in the organism, and the frequency of a mutation in a population, have a large influence on defining the pathogenic (or neutral) outcome of a mutation.

2.3.2 COMPUTATIONAL MODELS OF PATHOGENICITY

With a shift occurring in the medical domain, tending to the comprehensive understanding of the genetic etiology behind human disorders, it becomes hard to avoid the question of what is the function of each of the ∼25,000 human genes, and how does their genetic variation contribute to health and disease. Being able to computationally distinguish between pathogenic and neutral variants can aid this task significantly by identifying and prioritizing candidate mutations of patients [181], besides being cost and time advantageous when compared with the difficulty in attaining such knowledge experimentally. The process of creating computer models that are able to distinguish problematic variants from neutrals also gives itself insights into the mechanisms of disease. For example, if a model performs better when accounting for a particular characteristic change in a gene, then there is the likely chance that such characteristic is involved in the causative elements of the disease. However, the enormous complexity of genomics makes the creation of such disease-association models a difficult problem, as variants can affect biological function in numerous ways, and many external factors can also contribute to disease.

Nonetheless, with the sequencing of the human genome, and specially with the recent 1000 genomes project, a new era for human analysis and diagnosis began (a

progress partially depicted in Figure 2.4). Since these projects, numerous approaches

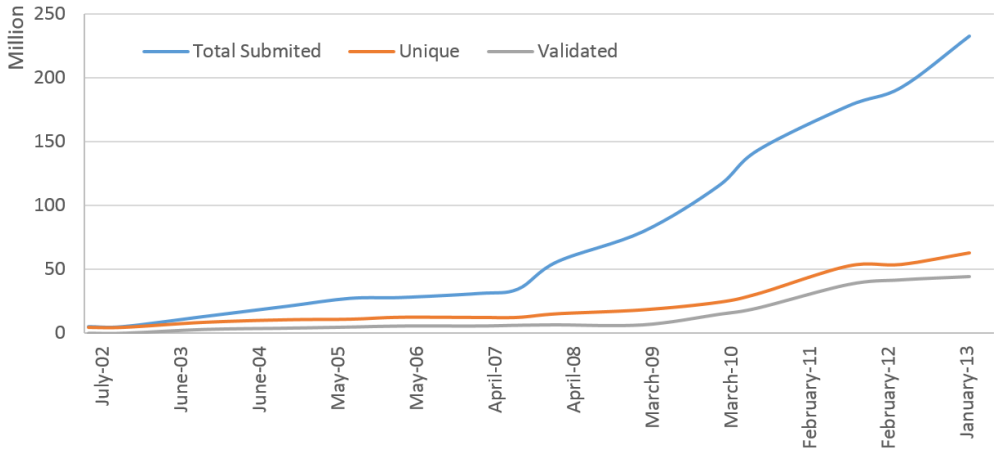


Figure 2.4 Growth of the SNPdb database. The total submitted (blue line) shows the raw number of human SNPs (SNVs with a frequency of 1% or more in the population) sent to the database. The orange line shows the number of variant clusters, i.e. different variants, and the gray line the ones that have been validated. A large change occurred since 2007/08, with the start of the 1000 genomes project.

haver emerged to crunch their data and extract significant information and predictions. However, on the topic of computational models of pathogenicity, attempts started as soon as variant data became available, even before the 1000 genomes project. The first solutions trying to tackle this challenge started in 2001 with the use of protein sequence homology to verify if the mutated site was conserved among homologous proteins in other species [128, 129]. The rationale behind this method, SIFT (Sorting Intolerant From Tolerant), is that if an amino acid is well conserved, it is likely to perform an important role in the protein, and therefore an amino acid change might be deleterious. Other authors expanded this concept and created rule-based methods also relying on the assessment of changes in the 3D structure of proteins at the same time [36, 177]. They measured, besides homology conservation, if a substitution was placed in an active binding site, affected interactions with ligands, lead to hydrophobicity changes in crucial core amino acids, among other structural characteristics. Another approach, Panther [180], combined homology conservation with annotated information comming from a constructed ontology, and introduced the use of machine learning by combining these attributes using Hidden Markov Models. By using a simple ontology, they were able to classify proteins by function, adding knowledge to their model of pathogenicity. In 2008, SNPs&GO combined the previous approaches using structural and physicochemical information with homology and

ontology [161]. The wide range of features, including information on functional alterations, role, and structure changes, along with the use of machine learning (Support Vector Machines), gave their method a high performance. Other methods focused in a specific functionality or goal, for example TANGO [59] was created to predict the formation of protein agglomerations, a trait resulting from genetic mutations that are specific to certain diseases such as Alzheimer's; or AUTO-MUTE, which was built to predict energetic variations in protein structures. As a result of their narrow domain specificity, they are able to achieve large predictive performances (for instance, 90% in AUTO-MUTE). Furthermore, several authors have created strategies to aggregate many of these methods into meta-prediction tools, under the hypothesis that their combination can achieve higher prediction performances than any of them alone, as is the case with META-SNP [32] and, recently, PredictSNP [18].

With the beginning of the 1000 genomes project, available data on human variation began rising and, as a result, more accurate and sophisticated methods could be developed and tested. The number of approaches and tools described in literature increased and gave the field a significant advance [3, 62, 82, 108, 162, 168, 169]. Table 2.3 depicts and compared the main methods described in literature.

Overall, the large majority of the methods rely on sequence conservation through homology verifications. As a result, these methods can only be applied to phylogenetically conserved regions, hence having poor coverage of the coding regions across the DNA [82], and do not take into consideration functional information [93]. Nonetheless, there are many indicators on which the methods rely to perform predictions, that can be roughly categorized into the following types of features:

- Amino acid features are those related with the mutated and neighboring amino acids, in terms of physicochemical properties, e.g. change in the hydrophobicity of the mutated amino acid.

- Structural features encompass characteristics related with the secondary and tertiary structures of the protein, such as the probabilities of the mutated amino acid pertaining to a solvent-accessible area of the protein, or pertaining to a domain known to have functional relevance, as well as predictions of three-dimensional changes.

- Homology features are related with the conservation of amino acids and nucleotides in other species, as well as the frequency with which pairs or trios of

	Year	AA features	Structural features	Homology	mRNA features	Role annotations	External factors	Trait-specific	Method	Accuracy
SIFT	2001	No	No	Yes	No	No	No	No	Rule-based	65%
Panther	2003	No	No	Yes	No	Yes	No	No	HMM	71%
TANGO	2004	Limited	Yes	No	No	No	Yes	Yes	Statistical	86%
PMUT	2005	Yes	Yes	Yes	No	No	No	No	Neural-network	80%
MAPP	2005	Limited	No	Yes	No	No	No	No	Calculated score	74%
PhD-SNP	2006	Limited	No	Yes	No	No	No	No	Decision tree+SVM	74%
SNAP	2007	Yes	Yes	Yes	No	No	No	No	Neural-network	72%
SNPs&GO	2008	Limited	Limited	Yes	No	Yes	No	No	SVM	82%
MutPred	2009	Limited	Yes	Yes	No	No	No	No	Random forest+SVM	81%
UMD-Predictor	2009	Yes	Limited	Yes	Limited	No	No	No	Aggregate function	80%
AUTO-MUTE	2010	No	Yes	No	No	No	Yes	Yes	Random forest+SVM	90%
MutationTaster	2010	Limited	Limited	Yes	Limited	No	No	No	Naïve Bayes	85%
FATHMM	2012	No	No	Yes	No	Yes	No	No	HMM	86%
Meta-SNP	2013	-	-	-	-	-	-	-	Random forest	79%
PolyPhen 2	2013	Limited	Yes	Yes	Limited	No	No	No	Naïve Bayes	70%
VAAST 2	2013	No	No	Yes	No	No	No	No	Likelihood ratio test	86%
PredictSNP	2014	-	-	-	-	-	-	-	Aggregate function	73%

Table 2.3 Pathogenicity prediction method comparison. Limited features translate that the approach has only a partial/incomplete use of that feature type. TANGO and AUTO-MUTE are trait-specific, as their goal is to predict protein energetic differences and protein agglomeration, and thus have a large accuracy in comparison with generalist approaches (whose average is 78%). META-SNP and PredictSNP are aggregators and therefore do not use features but rather the outcomes from other approaches. The use of mRNA features is very limited, and only present in three of the presented tools.

amino acids are found, e.g. values of the BLOSUM matrix, which reflect the substitution probabilities among amino acids.

- mRNA features account for characterizations of the coding sequence, such as changes in the energy of the secondary structure, frequency of the affected codon, presence of out-of-frame stop codons, or any other indicator described in section 2.2.1.

- Role annotations are features including external knowledge, such as that comming from the Gene Ontology, to annotate genes and proteins according to their function in the organism.

- External factors are the inclusion of variables influencing expression, pathogenicity, or any other trait, such as temperature and acidity pH levels.

Generally, approaches rely mostly in homology, suggesting different strategies to calculate conservation and prioritize variants according to it. Also, most tools are directed to nsSNVs, that is, variants where there was an amino acid alteration. However, according to the dbSNP database, only 59% of variants in coding regions are nsSNVs, and 36% are sSNVs. Moreover, it has been extensively shown that synonymous variants are implicated in the development of several diseases, rejecting the holding dogma that synonymous substitutions have neutral effects. Nonetheless, the use of codon and RNA-oriented evaluations is still largely ignored.

2.4 SUMMARY

Before the first sequencing of the human genome, the techniques to map and isolate the causes of genetic-associated diseases were slow-going and relied on labor-intensive tasks performed in the laboratory. For instance, efforts to identify the gene associated with Huntington's disease began in the late 1970s, but it was not found until 1993 [38]. Today, this job is much more streamlined, mostly due to the inclusion of information technologies in the process, and massive studies are now often created to identify the genetic locations responsible for specific traits and diseases.

In this chapter, I introduced the genetic background and advances that have been achieved in recent years on the topics of computational biology and protein biosynthesis research, specially regarding the methods that are used to evaluate and redesign gene coding sequences, and methods to predict the outcomes of genomic alterations in humans. I showed that the central dogma of molecular biology is largely responsible for the main processes of life, and that there is already a very good knowledge of the factors involved in its steps, specially in gene translation. These factors can be computationally measured and have broad applications, ranging from genomic analysis and gene redesign, to pathogenicity prediction.

The next chapters build on top of these concepts, using and extending the discussed concepts to tackle the gene optimization, manipulation and exploration challenges, as well as the variant pathogenicity prediction problem.

3 | GENE CHARACTERIZATION AND REDESIGN

Among the main goals of biotechnology is the ability to induce in a cell the production of a protein it would not create under normal circumstances. This sort of manipulation could, ideally, allow the creation of useful proteins, for example as therapeutics or industrial catalysts [196], or to further extend our understanding of living beings [5]. The creation or redesign of genes for different host species (heterologous expression), as discussed in chapter 2, is influenced by many variables, and such dependence may change widely among species, depending on their genetic machinery. Therefore, gene sequences must be engineered in order for the host cell to properly recognize its instructions and meet the redesign goals, such as, for example, maximizing the expression of a protein.

In this chapter I will discuss how to redesign genes to meet specific objectives based on the indicators previously acknowledged, and propose the methods developed for the optimization of that procedure.

3.1 OPTIMIZATION

The goal of redesigning a gene for heterologous expression is to be able to create the product of that gene in a target host, maintaining its functionality and, often, enhancing some of the characteristics of the process, such as the speed of translation, efficiency, or amount of created functional protein. Hence, the first rule to keep the conformation of the protein is to maintain the same sequence of amino acids: the primary structure of the protein. Any changes applied to the gene sequence must, thereby, be silent alterations, which involve changing codons by synonymous ones. The fact that several different codons can be used to encode each amino acid is the

essence of gene optimization.

Furthermore, as mentioned in the previous chapter, altering codons allows improving specific characteristics of the molecule and the decoding process. However, several issues arise when trying to improve two or more competing characteristics, or features that involve more than a single codon. For example, if the goal of a redesign is to maximize the amount of Guanine and Cytosine on a gene, while also minimizing codon usage, the two goals are concurrent because alterations are performed at the codon level and the trade-off between the two objectives might prevent obtaining an optimal result for both of them simultaneously. Another example occurs when considering the maximization of the codon pair bias of a codon sequence. A consecutive alteration of codons pairs that considers only a single pair at a time, in spite of previous alterations, will not achieve an optimal maximum codon pair bias in the whole gene.

The set of possible synonymous codon configurations is called the solution space, and the ease with which an algorithm traverses that space and looks for optimal configurations reflects the capability of that algorithm to optimize a gene. Moreover, the size of the solution space is equal to the amount of possible configurations, which is roughly 3^N for a gene with N codons, making it computationally unfeasible to completely explore. In the next sections I will overview some of the possible optimization algorithms to solve the afore mentioned problems.

3.1.1 DYNAMIC PROGRAMMING

Dynamic programming is an algorithmic design which allows solving complex optimization problems by identifying smaller subproblems and tackling them first, using the answers to aid resolving the larger goal. This is performed resorting to Bellman's principle of optimality:

> "An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision."
>
> – Bellman, 1952 [17]

The principle says that in an optimization procedure, subsequent states and decisions must be optimal, regardless of the first state and decision. The statement can be re-applied to any later subproblems, thus deconstructing the problem into

consecutive subproblems, i.e. having *optimal substructure*. Another characteristic of dynamic programming is relying on *overlapping subproblems*, that is, larger problems overlap with smaller ones such that the results from solved subproblems can be reused several times, for instance in recursive situations. This strategy is also known as memoization (or tabling), in which the results of computations for predetermined inputs are recorded, to avoid recalculations whenever the same computation occurs again, speeding up algorithms.

The codon pair bias optimization problem stands as a perfect example of a problem with *optimal substructure* and *overlapping subproblems*. To maximize (or minimize, for that matter) the codon pair bias (CPB) of a gene, it is guaranteed that any subsequence of the gene must also have maximum CPB, regardless of the remaining sequence. This logic can also be reapplied to any subsequences until the unit - a single codon pair - is reached, and must also be maximized. The property of *overlapping subproblems* can be found by the fact that solving the problem for a larger codon sequence (e.g. the whole gene) requires solving the problem for several subsequences, given that the problem relates to pairs of codons and therefore connecting every subproblem.

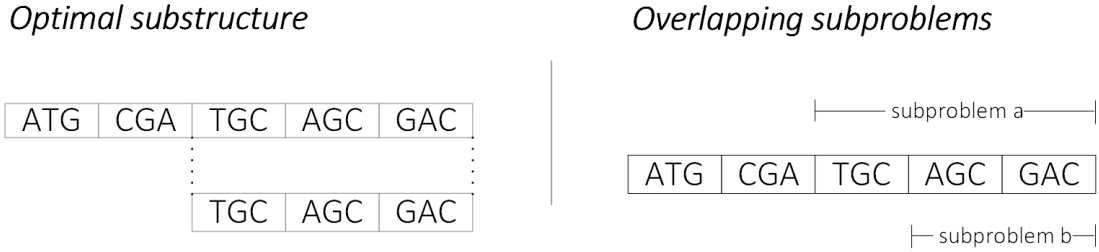


Figure 3.1 Dynamic programming requirements for the gene CPB optimization problem. Optimal substructure can be found by the fact that the solution to maximize the CPB of a whole gene requires finding the solutions for the maximization of a subsequence of the gene, hence decomposing the problem. Overlapping subproblems are observed by the fact that the CPB of the larger sequence depends on the CPB of a smaller subsequence (subproblem a), which in turn depends on an even smaller subsequence (subproblem b). Finding the solutions for the smaller subproblems allows speeding up the process of finding solutions for larger problems, since their computations can be cached.

Therefore, by exhibiting both principles, the CPB maximization problem can be dealt resorting to dynamic programming. A possible implementation to solve this problem involves going codon-by-codon, starting at one end of the gene, and recording for every possible synonymous in each position the best cumulative CPB score with

each synonymous of the previous position. Then backtracking every position and choosing the synonymous codon that yielded the best score. This simple algorithm guarantees yielding a synonymous gene with optimal maximum CPB.

Dynamic programming can also be used to achieve multiple purposes, i.e. multi-optimization, by evaluating problems and subproblems using objective aggregation functions. For instance, Pham et al. in 2004 used dynamic programming to optimize a gene sequence to improve codon usage while simultaneously avoiding certain specific sequence patterns, and include other [143]. In their approach, an aggregation function was created that evaluates sequences and sums a score for codon usage with a weighted score for the inclusion and exclusion of specific patterns. Such approach can be extended and scores from multiple evaluations, such as those presented in section 2.2.2, can be included in the aggregation function as long as Bellman's principle isn't violated.

Using dynamic programming avoids having to explore the solution space exhaustively, and enables the search for solutions to more complex problems that are not strictly single-codon oriented, as for example the problems of maximizing CPB or out-of-frame stop codons. However, dynamic programming is limited to problems depicting the principle of optimality. That is, problems that cannot be decomposed into smaller subproblems are hard to be solved by dynamic programming, or require reformulations. Another issue with dynamic programming is that the inputs must be discrete, or limited, in order for the algorithm to be able to efficiently store precomputed results (an issue derived from the overlapping subproblems requirement). Thus, problems with continuous inputs must be reformulated or cannot be solved with dynamic programming.

3.1.2 HEURISTICS

Heuristics are a class of problem solving techniques and algorithms designed to rapidly explore the solution space and yield solutions that are close to optimal, though not deterministically optimal. There are several advantages in using heuristics besides that of reduced time complexity. For once, heuristics allow a greater flexibility to be employed in abstractly complex combinatorial problems, when compared with deterministic algorithms, by not restraining themselves to specific classes of problems. Another advantage is that the search algorithm (the heuristic) is greatly decoupled from the evaluation functions, separating concerns and allowing the search to be possible with little knowledge of the problem. Also, by requiring only state evaluations,

heuristics generally avoid the need to have analytical formulations of the problem to be solved. This is largely useful when the goal to some problem is well defined, but a mathematical formulation of that goal is hard to attain.

Heuristics are often associated with stochastic optimization, which involve applying randomness in the algorithms to provoke non-deterministic behaviors. However, the behavior in itself is not random, but rather biased in the way it explores the solution space, guiding the process by surveying regions of the space that benefit the search for (near-)optimal candidate solutions. An example of such algorithm is the Hill Climbing (HC) method. In HC, an initial randomly-generated solution is created, and then random alterations are made iteratively to the solution, accepting only those that represent improvements. The algorithm terminates when no further improvements can be found. However, this algorithm suffers from an issue heavily fought by heuristic techniques: falling into local maxima solutions. In fact, the majority of problems present a non-smooth space of solutions, that is, the space shows regions that have solutions which are optimal in that region but are not the globally best solutions. By only accepting better solutions in each step, the HC algorithm easily gets stuck in local maxima, often returning poor results. Nonetheless, a possible implementation to optimize gene sequences would involve starting with a random synonymous gene, and in every step producing an alteration to a random codon by replacing it with a synonymous one (see Algorithm 3.1). If the new gene configuration represents a better solution to the goal, the method proceeds using that configuration, or otherwise the new solution should be discarded.

Algorithm 3.1 Hill Climbing algorithm to optimize genes

```
CurrentSolution ← Random synonymous sequence from original gene
Score ← Evaluate(CurrentSolution)
while End Criteria Not Met do
    NewSolution ← MakeAlterationInRandomCodon(CurrentSolution)
    if Evaluate(NewSolution) > Score then
        CurrentSolution ← NewSolution
        Score ← Evaluate(NewSolution)
    end if
end while
return CurrentSolution
```

Several variations and improvements to the Hill Climbing algorithm have been

proposed, for instance the Simulated Annealing[1] (SA) strategy [94]. SA is inspired by the process of annealing in metallurgy, where materials are heated and then slowly cooled in order to allow the atoms to find the best configurations, thereby improving the material resistance and reducing its defects [194]. SA brings the notion of heating and slowly cooling to the optimization landscape by employing a strategy that initially generates large random changes in the current solution, and in each cycle reduce the number of changes performed, while also easily accepting worse solutions at the beginning (thus avoiding local maxima) and slowly decreasing the acceptance rate until it only accepts better solutions near the end (see Figure 3.2). The HC algorithm is therefore improved by always accepting new solutions if they are better, but also accepting worst solutions according to a probability of acceptance derived from the current iteration and the difference in score achieved. Equation 7 shows a possible formulation of acceptance.

$$P_{acceptance} = e^{\frac{Evaluate(NewSolution) - Evaluate(CurrentSolution)}{\alpha \times \beta^{iteration}}}$$

Equation 7 Probability of accepting a new solution, as a function of the current iteration and the score of the new solution. i is the current iteration; β represents the cooling schedule,that is, how fast the temperature decreases, and is comprised between 0 and 1; and α a regulation factor to control the initial temperature.

SA has the advantage of being able to explore the solution space without becoming stuck in local maxima regions. Nonetheless, achieving good solutions requires a trade-off with the speed with which the temperature decreases: better solutions require larger computational time. The time required to obtain a near-optimal solution also increases with the size of solution space, which in gene redesign is equivalent to the number of possible permutations, and therefore dependent on the size of the gene. In fact, it has been shown that SA with appropriate cooling strategies will asymptotically converge to a global maximum [75]. A limitation of SA and HC involves the fact that they handle a single solution at any time, which reduces the depth with which the search is performed, and return only a single solution. When performing multi-objective optimizations, it is often the case that several different solutions are equally valid.

Evolutionary algorithms represent another class of heuristic-based methods. The computational methods in this class are based on premises from the biological evo-

[1] The SA method was actually proposed as an adaptation of the Methopolis-Hastings Monte Carlo algorithm. It is, nonetheless, an improvement to the simple HC.

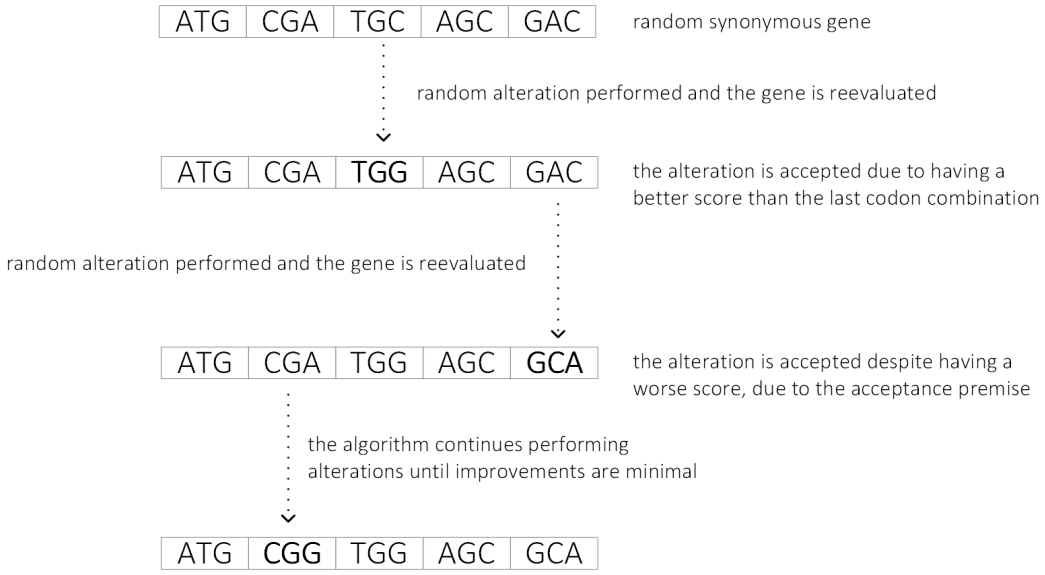


Figure 3.2 Simulated annealing algorithm example. An hypothetical gene is altered in each iteration, and the new gene is evaluated. Changes can be accepted or denied according to their score and the acceptance formula.

lution of species. In the theory of natural selection, evolution is achieved through a set of specific mechanisms, that allow genetic material to be passed on to offspring following simple rules that result in the adaptation of species to their surrounding environment. The same idea is used in evolutionary algorithms in order to adapt solutions to their goals, exploiting simplified properties of the theory of natural selection. These algorithms evolve a population (set) of candidate solutions through a cyclic procedure that seeks to gather and amplify positive variations and eliminate negative ones, by trial and error. Arguably the most well known methods in evolutionary algorithms are Genetic Algorithms (GA).

GA work by starting with a set of randomly generated candidate solutions and, in each iteration, evaluating each of the solutions, picking a sub-group of the solutions (selection), combine those solutions to generate new ones (crossover), apply some random changes to the new solutions (mutation), and then insert them into the population set. These steps are isolated and can be described as genetic operators, with many variations of them giving way to numerous variants of GA:

- The selection operator can be performed by a *fitness proportionate selection* (also known as roulette-wheel), where only a fraction of the population is ran-

domly selected for the next step with a probability proportional to their score. Other methods for selection include for instance choosing only a top scoring fraction of the population (*truncation selection*).

- Crossover can be achieved by picking two random candidates from the selection phase, and creating one or more new solutions by combining them. One strategy to perform combination, is to pick a random position in the solutions and swap the solution vectors after and before that position. Uniform crossover is another strategy which involves swapping a random fraction of the solutions.

- Finally, in the mutation operator, one or more random changes are created in the new solutions, to ensure genetic variability in the population. This is normally performed by altering a fixed fraction of the new solutions.

The new solutions can be inserted into the population set by excluding the worst fit, or replacing the parent solutions, though there are also several other strategies to this step. According to the theory of evolution, facilitating the crossover of solutions with higher scores eventually leads to new solutions further adapted to improve the score.

GA have the advantage of considering multiple possibilities simultaneously, thus conferring the method with a deeper perspective of the solution space. Dealing with multiple candidates also allows finding multiple final solutions to the same problem, which is specially useful under multi-optimization schemes. Moreover, by being constructed from several operators, the algorithm is flexible to adapt to different problems. However, the number of operators also comes with the drawback of requiring a large amount of configurations and parameters that may be fine-tunned, such as the fraction of solutions that are selected for crossover, or the number of mutations of perform on a newly created solution.

3.1.3 SPACE CONVEXITY PROBLEM

Global optimization algorithms such as those presented before can be used to perform searches that find minima or maxima for a single function with respect to a goal. However, most instances of real problems involve several simultaneous constraints, that is, having more than one criterion to optimize, called multi-objective problems [194].

In the context of genetics, redesigning a gene for heterologous expression might require obeying to several criteria in order for the gene to correctly synthesize its product and eventually enhance its expression. For instance, one might want to harmonize the codon usage of the gene to guarantee the reliability of translation, while simultaneously avoiding specific sites that might be deleterious to the mRNA, and also maximize the amount of hidden stop codons to avoid resource consumption when errors occur.

Heuristics offer a possible solution to this issue since they rely on external evaluation functions that ultimately determine the goal of an optimization. The advantage is that it is easier to create evaluating functions than creating analytical formulations of the problem to be solved. This is specially true in the context of genetics, where our understanding is related to mechanisms and observations, rather than formulas. For instance, it is trivial to evaluate the amount of out-of-frame stop codons in a gene, but it is far more complex to redesign a gene to deterministically maximize their presence (it requires using, for example, a dynamic algorithm such as that presented in section 3.1.1). Heuristics also ease the process of joining several evaluation functions, since the evaluation is completely decoupled from the optimization procedure, and therefore several requirements can be aggregated in a single evaluation function. After defining individual constraint and goal functions, there are several strategies to create an aggregate objective function. The most common and simple method is to perform a linear aggregation of the objective functions, allowing the custom definition of importance (weight) for each objective (see Equation 8). This approach also allows minimizing or maximizing individual functions withing the set of functions of the problem, by assigning negative or positive weights.

$$F(s) = [f_1(s), f_2(s), ..., f_n(s)]$$

$$AOF(s) = \sum_{\forall f_i \in F} w_i f_i(s)$$

Equation 8 An example aggregate objective function (AOF) built from the weighted sum (linear aggregation) of every evaluation function f. An heuristic uses the AOF to evaluate a candidate solution. This strategy also permits having some functions being minimized while others are being maximized.

One of the drawbacks with this approach is the difficulty in handling problems where individual objective functions rise and fall asymptotically with different cadence, making the contribution of some of the functions negligible compared to the

ones with larger growths [49]. This is hardly the issue with gene redesign problems, since the limited genetic mechanisms and the boundaries of the solution space make evaluation functions generally have upper and lower bounds with limited growths. Nonetheless, another limitation with this approach is related with the structural shape of the solution space: the impossibility to return solutions if the space is not convex.

Ultimately, the goal of the aggregation function is to find a solution from the solution space that maximizes (or minimizes) all the objective functions. In a well-defined problem, however, the objectives are conflicting with each other, and the goal of the aggregation function becomes finding the optimal trade-off between the objectives. For example, imagine redesigning a gene to maximize its average codon usage while also maximizing its CG content. Both objectives work at the codon level, and choosing a synonymous codon in each position to benefit one of the objectives could easily conflict with the optimal choice for the other objective.

Having conflicting goals, somehow, transmits the erroneous idea that the best compromise is related to the spacial distance between candidate solutions and the utopian or ideal solution, that is, the closer (in Euclidean distance) a candidate solution is to the utopian solution the better (see Figure 3.3).

However, this assumption only holds in convex solution spaces [122]. In non-convex spaces, such as that depicted in Figure 3.3, many solutions are posed to be equally fit despite being abstractly distant from the ideal solution. Therefore the possibility of having non-convex spaces in a problem nullifies the use of a linear aggregation function, since it is a direct extrapolation of a Manhattan distance and will only work in convex parameters. Proving the convexity of a solution space is hard, and requires analytical formulations of the problem, which is one of the reasons that leads to choosing heuristics (to avoid formulations).

Vilfredo Pareto, who bootstrapped the mathematical foundations for multi-objective optimizations with conflicting criteria, studied this issue and defined as *Pareto front* the set of solutions that can be reached by trading-off conflicting objectives in an optimal manner (bold line in Figure 3.3) [194]. The solutions in the *Pareto front* are equally fit, that is, no solution is better than another as to all the objectives, and represent the best solutions to the problem. The *Pareto front* can be non-convex, and its concavity can be abstractly large, that is, solutions within it can be unequally distant (Euclidean distance) from the ideal solution.

One of the strategies to find *Pareto optimal* solutions (i.e. those located in the *Pareto front*) is to use improved metrics to the simple linear aggregation, which is

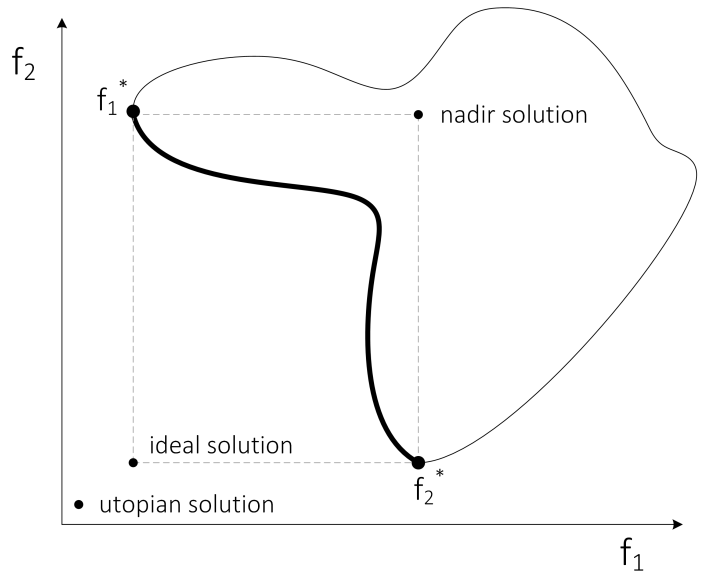


Figure 3.3 Non-convex solution space and its Pareto front. The solution space of a hypothetical problem is shown, depicting two functions f_1 and f_2 to be minimized concurrently. The Pareto front solutions are shown in the thick black region. The minimum of each function is represented by f_1^* and f_2^* respectively. The nadir solution is the realistic upper bound to the problem. The Pareto front is enclosed between the nadir and ideal solutions, though none of them has to actually exist.

a weighted form of the Manhattan distance [116]. In fact, both the Euclidean and Manhattan distances are specific cases of a more generalized formulation called the Minkowski distance (see Equation 9).

$$\text{Mink} = \sqrt[p]{\sum_{i=1}^{n} |x_i - y_i|^p}$$

Equation 9 Minkowski distance between points x_i and y_i. When $p = 1$, the equation becomes the Manhattan distance, and with $p = 2$ it becomes the Euclidean distance. Another common variation is with $p = \infty$, known as the Chebyshev distance.

When using an infinite p, the Minkowski distance becomes the Chebyshev distance, which basically reflects the maximum value among all objective functions: $\max_{\forall f_i \in F}(f_i)$. The Chebyshev distance has the advantage of guaranteeing to be able to retrieve all *Pareto optimal* solutions, even in non-convex settings [51, 91]. To use it, the AOF should be redefined to be the Chebyshev distance from a candidate solution in the heuristic to the ideal solution (granted it can be calculated). Though it does not

guarantee returning all solutions in the *Pareto front*, it does have the ability to return them, as opposed to a Manhattan or Euclidean distance. Several other methods have been presented to overcome the problem of capturing non-convex portions of the *Pareto front* [116], such as the exponential weighted criterion [12] and physical programming [121].

Despite the mathematical resolution that allows finding optimal solutions, there is also an algorithmic strategy to retrieve them, even when using linear aggregations in non-convex spaces. When resorting to heuristics, there is an advantage in terms of space exploration, since the algorithm visits and evaluates numerous solutions. Considering this, an archive of *Pareto front* points has been suggested which enables the tracking of visited solutions that are *Pareto optimal*, independently of the aggregation function [97, 185]. This is possible because assessing if a candidate solution dominates all other potentially optimal solutions visited so far is a feasible task which requires only a few evaluations and comparisons. A solution dominates another when it is better in every goal (objective function). The algorithm works by confronting each visited candidate solution with an archive of solutions that are currently believed to be the *Pareto optimal* solutions. When a new candidate solution is not dominated by any solution in the archive, or is found to be better (dominates) some solutions in the archive, it replaces those solutions and is kept in the archive. The step is repeated until the heuristic terminates, guaranteeing that at the end only non-dominated solutions (potentially belonging to the *Pareto front*) are in the archive.

3.2 METHODS

3.2.1 ALGORITHMS

Several algorithms were developed to evaluate genes regarding each gene expression and quality indicator. Generally speaking, the algorithms take an mRNA sequence and produce a numeric value representing an evaluation of the indicator over that sequence. For instance, algorithm 3.2 shows the pseudo-code for evaluating a gene regarding its codon pair bias.

Most algorithms that consider two consecutive codons at a time, such as the hidden stop codons evaluation, which counts the presence of out-of-frame stop codons between every two codons, have a similar structure to the calculation of CPB. Moreover, we devised evaluation algorithms for all the indicators in Chapter 2 (see the

Algorithm 3.2 Gene CPB evaluation

$N \leftarrow$ length of gene in codons
for $i = 1$ to $N - 1$ **do**
  $codon_a \leftarrow$ codon at position i
  $codon_b \leftarrow$ codon at position $i + 1$
  $\text{CPS} = ln\left(\frac{F_{a,b}}{\frac{F_a F_b}{F_x F_y} F_{x,y}}\right)$ ▷ As explained in Chapter 2
  $accumulator \leftarrow accumulator + CPS$
end for
return $\frac{accumulator}{N}$

details of the developed algorithms in Appendix A).

As mentioned before, the advantage of having evaluation rather than redesign algorithms for each expression and quality indicator is two-fold: first, it allows evaluations to be combined in aggregation functions and used in different optimization strategies; and second, creating specialized algorithms to redesign genes according to a set of observable rules is abstractly more complex than creating an algorithm to evaluate a gene. The latter can be seen, for instance, when trying to deterministically achieve optimal CPB using a dynamic programming approach (algorithm 3.3), where the complexity of the redesign strategy is substantial.

Nonetheless, in terms of efficiency for optimization procedures where only a single indicator is used, and to accurately calculate ideal and nadir boundaries, using specialized deterministic algorithms is advantageous.

When heuristics were needed, however, we used simulated annealing for fast results and genetic algorithms for a more extensive overview of *Pareto optimal* solutions. Moreover, we employed the Pareto archive strategy when using GA to help avoiding the problem of non-convex solution spaces. Each generated candidate solution is evaluated in order to keep track of *Pareto front* points that traverse the population. We defined gene dominance as follows (assuming a maximization problem) [67]

[**synonymous dominance**] g_i dominates g_j if and only if $f_k(g_i) \geq f_k(g_j), \forall k$ and there is at least one k such that $f_k(g_i) > f_k(g_j)$

Here g_i and g_j are the synonymous genes being compared and f_k refers to any evaluation function being considered in the optimization procedure.

Algorithm 3.3 Gene CPB redesign using dynamic programming

```
# First part: memoization
CPS[1][syn] ← 0, ∀syn in position 1                    ▷ Initialize CPS array
N ←length of gene in codons
for i = 2 to N do
    for each synonymous syn_a in position i do
        CPS[i][syn_a] ← max{CPS[i − 1][syn_b] + CPS(syn_a, syn_b)}, ∀syn_b
    end for
end for

# Second part: backtracking
M ← max{CPS[N][syn]}, ∀syn
RedesignedGene[N] ← c ← j, such that CPS[N][j] = M
for i = N − 1 to 1 do
    for each synonymous syn in position i do
        if CPS[i][syn] + CPS(syn, c) = M then
            RedesignedGene[i] ← c ← syn
            M ← M − CPS(syn, c)
            continue to next position i
        end if
    end for
end for
return RedesignedGene
```

We developed a GA using a Pareto archive, similar to the Strength Pareto Evolutionary Algorithm [204], and which is shown in algorithm 3.4.

We also use a fitness proportionate selection as the section operator, thus choosing candidate genes from the population with a probability proportional to its AOF value. This choice has been shown to be more advantageous as it maintains variety in the population. The crossover was performed by swapping codons of the same positions, between the parents, in a random fashion, i.e. each position of the new gene is randomly selected from one of the parents. It uses two parents to yield two new candidate solutions. The mutations applied to each new candidate solution are performed by randomly changing a fixed proportion of the gene by synonymous codons. To decelerate the rate of mutations as the genetic algorithm approaches the

Algorithm 3.4 Gene redesign using a genetic algorithm

```
Population ← Random sequences, synonymous to original gene
Archive ← GetNonDominated(Population)
repeat
    Parents ← Selection(Population)
    Children ← Crossover(Parents)
    Children ← Mutation(Children)
    if AOF(Children) > AOF(Parents) then
        ReplaceParentsByChildren()
    end if

    # Pareto archive maintenance
    NewArchive ← Union(Archive, Children)
    Archive ← GetNonDominated(NewArchive)
until EndCondition
return Population[BestIndividual]
```

end, we used equation 10 to control the probability of applying a mutation.

$$P_{mutation} = \frac{it_{max} - it_{current}}{k \times it_{max}}$$

Equation 10 Formulation of the probability of mutating a codon. Depending on the current iteration $it_{current}$, the rate of mutations decreases linearly as the algorithm proceeds. The initial mutation rate is controlled by k.

Because k controls the mutation rate, it also controls the rhythm of the evolution in the heuristic. Larger k leads to fewer changes in the offspring, promoting the selection of local maxima solutions.

For evaluating the fitness of each candidate solution, we used the Chebyshev distance between the solution and the ideal solution. Furthermore, to guarantee that every objective function is treated equally and the difference in scale of functions does not introduce bias in the aggregation function, we normalize each objective function using its ideal and nadir solutions (Equation 11).

The ideal solution is calculated by performing individual optimizations for each goal function, using deterministic algorithms. By finding solutions that are guaranteed to be individually optimal for each objective function, we also guarantee its

$$f_i^{norm} = \frac{f_i - z_i^{nad}}{z_i^{ideal} - z_i^{nad}}$$

Equation 11 Normalization of arbitrary goal function f_i by rescaling it using values from the nadir and ideal vectors.

presence in the *Pareto front*, and are able to retrieve values for the ideal solution. Since nadir solutions are hard to find or calculate, hypothetical upper bound values (such as the individual maximum for each function) are used to replace them. The same is performed when the ideal solution cannot be easily attained. The normalized functions become scaled to the range $[0, 1]$.

3.3 RESULTS

3.3.1 OPTIMIZATION RESULTS

We assessed the viability of redesigning genes through heuristic optimization by testing the algorithms in 40 different genes from several species, in two separate experiments (depicted in Figure 3.4). In the first experiment 20 genes were redesigned to obtain codon configurations that maximize their Codon Pair Bias and simultaneously maximize codon usage (RSCU). In the second experiment, we randomly selected another 20 genes, and added the goal of minimizing the amount of out-of-frame stop codons. This increases the difficulty of the problem substantially due to the existence of several competing objectives. Experiments were performed using both simulated annealing and genetic algorithms, though the difference between results (final solution in SA, and best candidate in the population in the last iteration of the GA) was residual and not statistically significant ($p > 0.1$). From every sample we first measured the score of the original gene and then optimized the genes with each goal individually in order to obtain the values that constitute an hypothetical ideal solution. For example, in the first experiment, prior to the multi-optimization, we used the dynamic programming approach to deterministically obtain the score of the codon combination with the highest CPB, and also ran an algorithm to obtain the best possible RSCU. The individual scores were aggregated using a simple average. We then performed the multi-optimizations and collected the score of the final result as a percentage of the AOF of the ideal score.

In experiment 1 we observed improvements averaging 31 percentage points, while

in experiment 2 the average improvement was 24 percentage points. No individual objective attained a worse score than the original. Also, every optimization result was on average 12% distant from the hypothetical ideal solution.

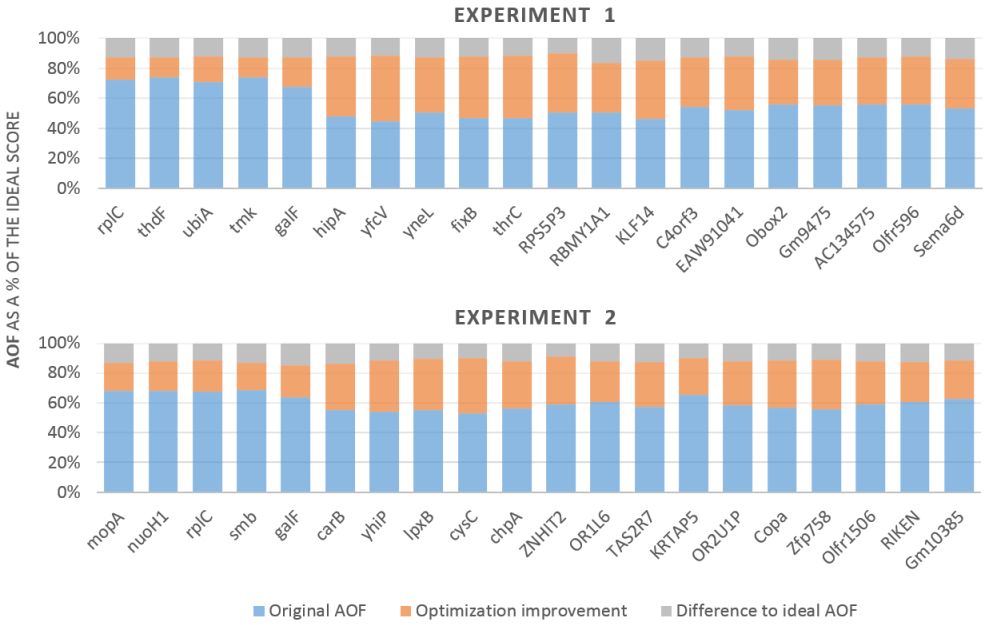


Figure 3.4 Results of gene redesign experiment using SA. Results are shown for each gene, as the percentage of the ideal solution score. The blue bar represents the score of the original gene (prior to optimization), the orange bar represents the achieved improvement, and the gray bar represents the unachieved difference to the ideal solution score.

We also observed the ability of the genetic algorithm in exploring the solution space more deeply and, with the aid of the Pareto archive, being able to retrieve *Pareto optimal* solutions (Figure 3.5). Simulated annealing, on the other hand, obtains good solutions in only a fraction of the time, but also only finds a fraction of the front.

3.3.2 GENE REDESIGN PLATFORM

In order to advance and facilitate synthetic gene design and analysis, we developed a tool that integrates expression and quality indicators, along with the synthetic gene redesign techniques explored previously. The tool, Eugene, was created to allow the synthetic manipulation and analysis of genes through a researcher-friendly interface (Figure 3.6).

Eugene's capabilities can be divided in two blocks: data gathering and gene optimization. In the first block, a set of features allows the retrieval of information about

Figure 3.5 GA and SA redesigning a gene for RSCU and CPB simultaneously. The solutions for every iteration are plotted for each of the optimization algorithms. Simulated annealing considers a single solution in each iteration, while the genetic algorithm uses a population. This fact confers the genetic algorithm a better ability to retrieve *Pareto optimal* solutions.

a gene from several known online sources, offline tools, and statistical calculations. In the second block, the tool allows redesigning a gene according to the already discussed factors that influence gene expression and quality, often resorting to the data gathered in the first block. These are discussed in the following sections.

Data aggregation

A large amount of information about genes and genomes can be considered (or is needed) for gene *in silico* analysis. For instance, it is common to evaluate the protein's secondary and tertiary structures to map the codon sequence to these structures. Unraveling which codons are responsible for the correct folding of a polypeptide might be essential when redesigning genes. Another clue for identifying important regions of a gene is given by its level of conservation among orthologs. Moreover, other forms of information such as the set of highly expressed genes for a genome are required in order to calculate some indicators, for example CAI values. Therefore, Eugene

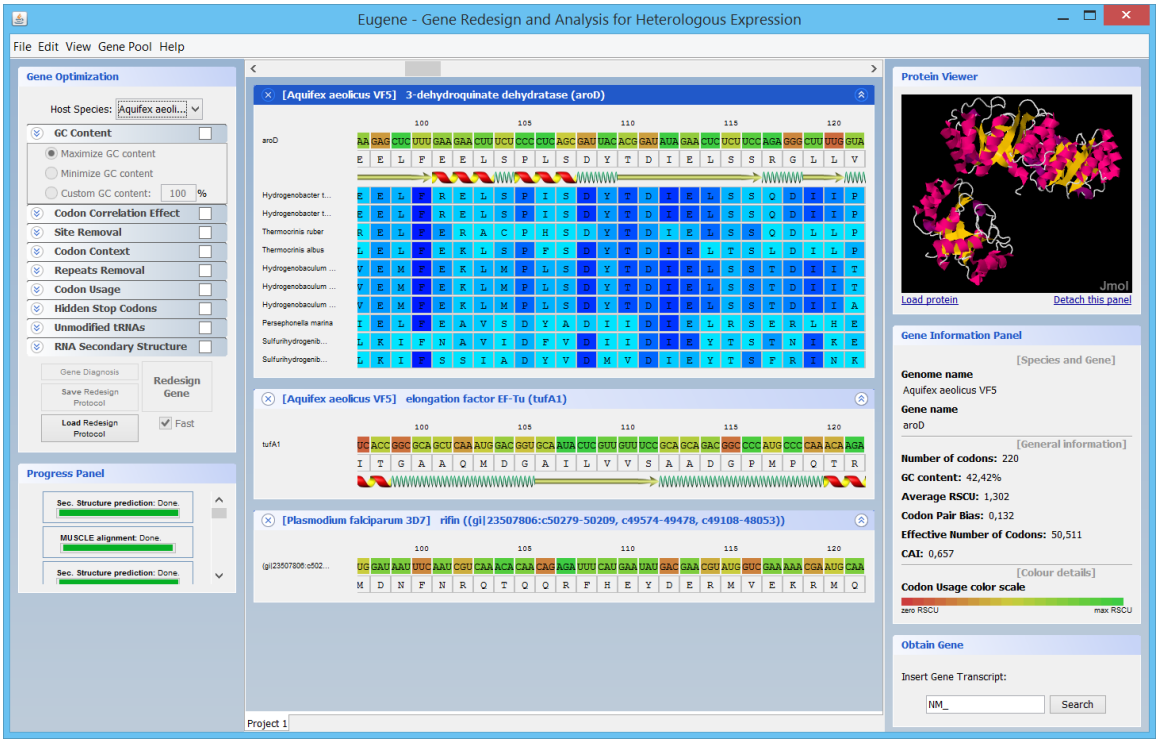


Figure 3.6 Screenshot of Eugene's interface. Three vertical zones correspond to: the redesign panel (left), where expression indicators are available to control; the gene workspace (center), where genes are uploaded to and analyzed; the informative panels (right), where details about the gene are shown, such as values of indicators and the retrieved tertiary structure of the resulting protein.

encompasses a series of features related to gathering data useful both to the analysis of the gene and to the calculation of several of the redesign indicators. Data is obtained or computed from several sources, such as online databases and offline tools (Figure 3.7). A more detailed description follows.

- Eugene has the ability to read the most common genome formats (FASTA and GenBank), and calculates the frequency of each codon, pair of codons and amino acid in the coding regions of a genome immediately after reading it. This information allows calculating RSCU and CPB, and several other indicators that require codon usage data.

- To correctly retrieve further data regarding a gene uploaded to the workspace, Eugene seeks to identify the gene using two strategies. The first is processing gene annotations that are normally present along with the gene's DNA sequence in FASTA and GenBank formats, extracting any database identifiers.

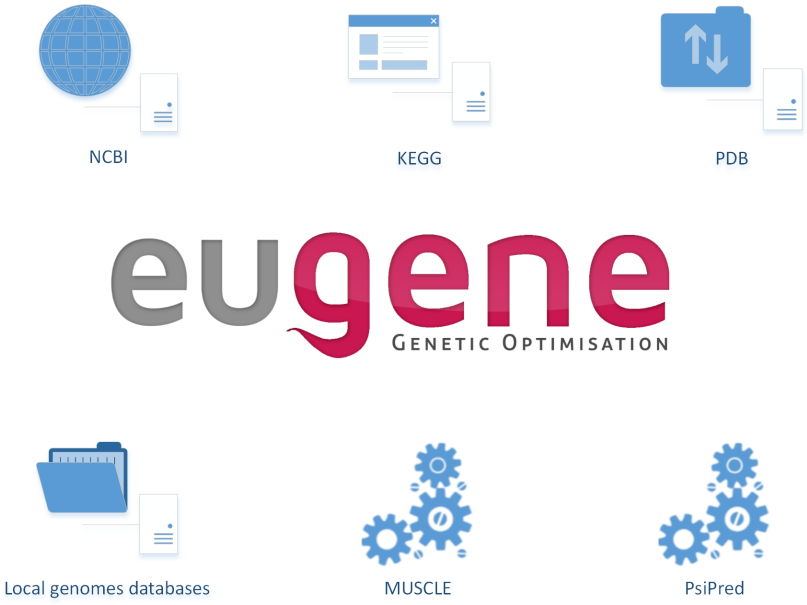


Figure 3.7 Eugene's use of external resources. On top, online databases, which also include online tools like BLAST. On the bottom, offline resources such as the MUSCLE and PsiPred tools used by Eugene, and the databases of genomes in the user's computer, which is the source of genes.

The identifiers are used to access NCBI and obtain names for the gene, genome and resulting protein. If the first strategy fails, a second one uses the gene codon sequence to search for similar sequences in NCBI's online database. The search is performed using BLAST [88], and the best match with similarity over 95% is selected. If a match is found, the same information is retrieved (gene, protein and genome names).

- To obtain the tertiary structure of the protein, a BLAST is also performed using the Protein Data Bank web services and the best and most complete match is selected. The protein 3D structure is visually mapped to the codon sequence, easing the understanding of which codons decode to which regions of the protein.

- The names and other retrieved meta-data are then used by Eugene to obtain orthologs for the gene. Orthologs are genes in different species that descend from a same gene in a common ancestor. Analyzing orthologs allows inferring a great deal of information, for instance, one can verify which regions of a gene are conserved among orthologs to deduce regions that are functionally

relevant, since they were maintained along several evolutionary lines. To retrieve orthologs, Eugene contacts KEGG web services [135] using information about the gene identification retrieved previously.

- Orthologs are seamlessly aligned with the gene using MUSCLE [54], and colored in the interface according the degree of conservation of each amino acid, allowing a visible assessment of conserved and functional regions.

- Furthermore, calculating the CAI indicator requires a set of highly expressed genes for the genome that will be hosting the redesigned gene. To obtain those genes, we manually curated a set of highly expressed genes for a single species (*E.coli*), and programmed Eugene to search KEGG for orthologs to those genes in the target species. With this strategy we can retrieve a set of highly expressed genes for virtually any species available in KEGG.

- Moreover, the protein secondary structure is also seamlessly calculated using PsiPred [119] and presented in Eugene next to the amino acid sequence.

- When working with specific genes, a functionality allows fetching an mRNA sequence by indicating its NCBI transcript ID. NCBI web services are contacted to retrieve the gene.

All retrieved and calculated information is always displayed. That includes, for each gene, its size in codons, CAI, G+C content, N_c effective codons, average RSCU, CPB, protein primary, secondary and tertiary structures, orthologs, and corresponding names.

GENE REDESIGN

The main functionality of EuGene is optimizing synthetic codon sequences according to user-customized indicators. A set of nine redesign approaches is available for both optimizing and analyzing the gene: G+C content, codon correlation effect, removal of deleterious sites (such as Shine-Dalgarno), codon context (CPB), control of repetitions, codon usage (RSCU, CAI and rare codons), hidden stop codons, unmodified tRNAs, and mRNA secondary structure.

Accordingly, all modifications are performed without changing the resulting native amino acid sequence. Moreover, changes are controlled by the genetic code and codon usage/context tables of the target host species and, therefore, if the host species

is non-native, the protein primary structure is always maintained and all redesign considerations are made using the statistical information of the host. This allows, for instance, harmonizing the codon usage of a gene to a heterologous host, allowing a researcher to express a gene in a different species while maintaining its original protein sequence and codon usage signature. The target host can be one whose genome data (FASTA or GenBank file) was uploaded into the application.

EuGene allows simultaneous optimization of any redesign approaches by using two multi-objective optimization techniques: a genetic algorithm and simulated annealing. Both strategies aim at finding the overall best gene candidate for the redesign objectives. However, simulated annealing is faster in finding a single optimal codon configuration and is, therefore, ideal for experiments that need fast results. For example, Figure 3.8 depicts a redesign of the human gene AXIN2 by maximizing its codon usage using simulated annealing.

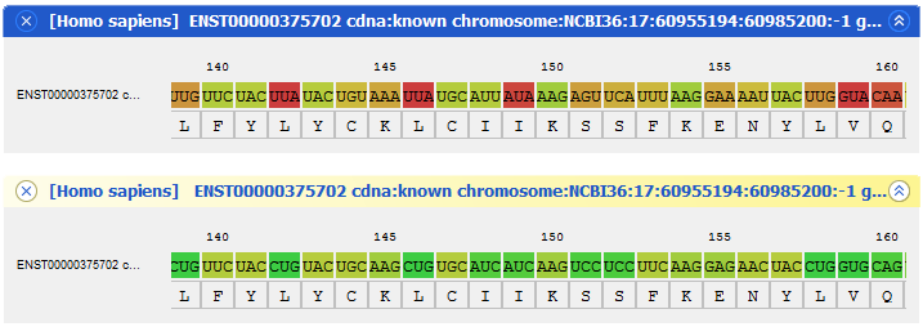


Figure 3.8 Screenshot of codon usage redesign in Eugene. A block of codons from the AXIN2 original human gene is shown above (codons 139 to 160), along with its corresponding amino acid chain. Codons are colored according to RSCU values, ranging from red infrequent codons to green frequent codons. The same gene block is shown below after a codon usage maximization redesign, showing a much greener variation (higher RSCU).

When a deeper overview is required of several equivalent solutions that trade-off the redesign goals (i.e. Pareto optimal solutions) a genetic algorithm optimization may be used. Using the genetic algorithm yields to the user a list of non-dominated solutions found to constitute the Pareto front, so he can adequately choose one. A chart is also displayed plotting those solutions using any two objective functions among those chosen to perform the redesign (Figure 3.9).

Eugene was created so new gene expression indicators can be easily added to the software, by using a plug-in system. Such feature is a requirement in a domain constantly growing its knowledge.

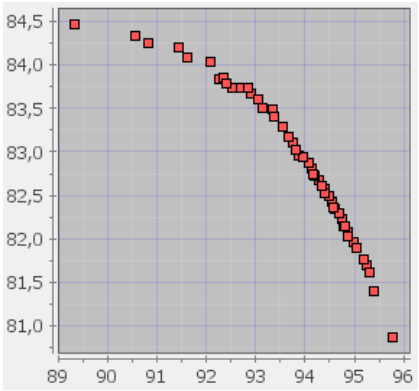


Figure 3.9 Screenshot of the Pareto front chart in Eugene. Every square represents a non-dominated solution. The vertical axis represents the goal of maximizing codon usage, and the horizontal axis maximizing the number of hidden stop codons. Note that several of the plotted solutions would have not been retrieved if it wasn't for the Pareto archive, considering the non-convex shape of the front.

3.4 CONCLUSIONS

As the technological capabilities to observe living beings at microscopical level increase, literature becomes flooded with information regarding their functioning and, in the past decades, the functioning of all processes gravitating around the central dogma of molecular biology. Protein biosynthesis is an intricate procedure, involving many cell actors, variables and influencing factors. As new roles, components and cell activities are discovered, the knowledge of the biosynthesis process is also updated and, as a result, new improved forms of controlling the expression of genes can be created. It is up to computational biologists to include the newly created knowledge into existing systems and tools. Streamlining this procedure is hard due to the arbitrary complexity that such knowledge might bear.

In this chapter I focused on the problem of including the acquired and continuously growing knowledge regarding protein synthesis into a single system with the purpose of redesigning genes. Being able to redesign genes enables the production of proteins in foreign hosts, control over the amount of synthesized products, and rational design of new polypeptides; applications range from efficient vaccine production to the creation of industrial catalysts, crossing through the study of the mechanisms of new species and diseases.

The largest barrier to the integration of this kind of knowledge lies precisely in its

arbitrary complexity and the range of subjects it can encompass. As shown, expression indicators can rely on disperse blocks of information such as codon frequencies, mRNA folding energies, presence of specific motifs, conservation among species, etc. Moreover, analytical formulations for each expression indicator can be extremely complex and time-consuming, whereas on the other hand evaluations are commonly trivial to calculate. We tackled this issue by creating a mandatory programming interface to any expression indicator to be included in our strategy, in which a numeric evaluation is independently performed and normalized according to an hypothetical best and worst case (ideal and nadir solutions). This strategy enables the aggregation of virtually any approach to evaluate genes that can be quantitatively measured.

Another obstacle is related to the combined redesign of genes. Given the unknown complexity underlying evaluation functions, the optimal solutions to a redesign problem can assume random shapes, many of which may be non-convex. The non-convexity of the *Pareto front* is an issue intrinsic to multi-optimization challenges, and which we tackled by using a specific case of the Minkowski formulation: the Chebyshev distance.

We also explored optimization itself, by considering two solution-space exploration techniques to permute synonymous codons: simulated annealing and genetic algorithms. On one hand, SA permits rapidly traveling through the solution space to find good or even optimal solutions. On the other hand, multi-redesign problems often create conflicting goals where a trade-off is observed in the *Pareto front*, and therefore several solutions become equivalent. We developed a genetic algorithm with a Pareto archive that maintains a set of non-dominated sequences while redesigning a gene, offering the chance to choose a gene from a set of equivalent solutions at the end of the optimization.

Furthermore, to enable the access to and extend the use of our methods, we further continued the development of an existing software application for gene manipulation, Eugene [65], which already allowed the integration of redesign algorithms through a plug-in system, and performed genetic algorithm optimizations. We integrated our methodologies in Eugene by implementing simulated-annealing, gene dominance and a pareto archive, solution-space exploration techniques, and several gene evaluation algorithms. Eugene is available at bioinformatics.ua.pt/eugene.

4 | mRNA STRUCTURE CONTROL[1]

Among the many factors that influence gene translation, the role of mRNA secondary structure has long been shown to be of major importance [50, 76, 102]. For instance, regulation of gene expression is highly dependent on the formation of stable structures by nucleotide pairing in the mRNA strand. This is especially true when the structures encompass translation initiation regions, hence hampering the start of the decoding process [104, 202].

We discussed in previous chapters strategies to evaluate gene characteristics, mainly based on codons or small regions of the mRNA. However, evaluating the potential formation of secondary structures is a much more complex subject, target of much research. Besides encompassing the evaluation of the whole mRNA sequence simultaneously, even non-coding regions, performing structural analysis takes a considerable amount of time with current methodologies, which makes its use in heuristics unfeasible.

In this chapter we discuss how to simplify the evaluation of mRNA secondary structures, at the cost of losing some of the accuracy of the method and the amount of information retained from calculations (for instance, the visual secondary structure itself), by approximating a trivial measurement function to the results of accurate methods. We also show how this technique enables the use of heuristics to control the secondary structure of a gene resorting to codon permutations, similar to what was presented in Chapter 3.

[1] This chapter was largely based in the publication *mRNA secondary structure optimization using a correlated stem-loop prediction*, Nucleic Acids Research, 2013 [68].

4.1 RNA STRUCTURES AND FREE ENERGY

The formation of stem-loops and more complex structures occurs upon RNA folding on itself causing secondary and tertiary nucleotide interactions, whose stability is dependent on the nucleotides involved and the length of the interacting domains (see Figure 4.1). The strength of two paired bases is largely determined by the number of hydrogen bonds that connect the nucleotides: guanine-cytosine pairs share three hydrogen bonds, and adenine-uracil pairs have only two; the wobble base-pair guanine-uracil also share two hydrogen bonds. Longer paired zones and stronger paired zones tend to be more stable, and therefore have higher melting temperatures, preventing the ribosome from breaking the pairing and proceeding translation [102, 174].

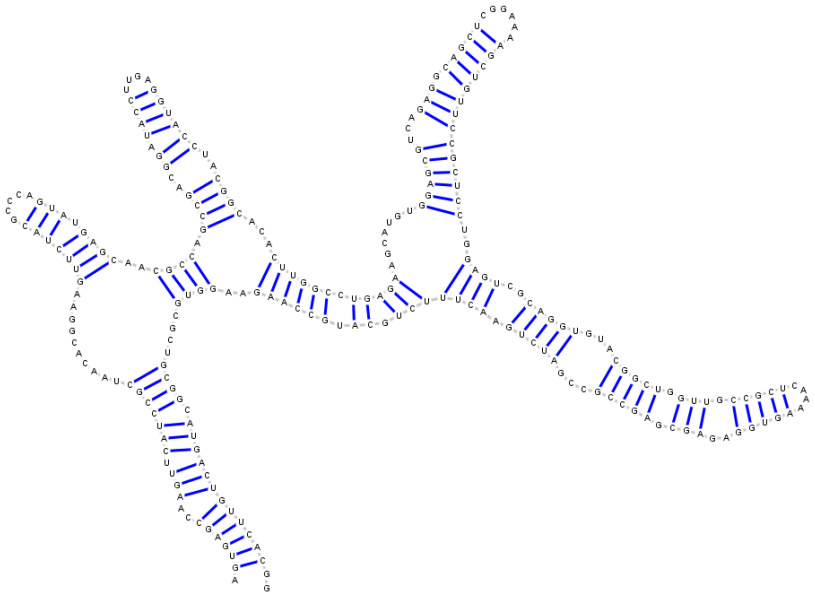


Figure 4.1 Secondary structure of an RNA sequence. The nucleotide sequence is shown as a string, where paired non-consecutive nucleotides are shown with blue lines. The paired nucleotides create the secondary structure of the mRNA (the primary structure being the sequence itself).

Several recent studies have demonstrated that manipulating RNA sequences to avoid secondary structures has a substantial impact on gene expression. Studer and Joseph, in 2006, performed experiments where they changed several mRNA sequences to control the presence and strength of secondary structures near translation initiation sites, and showed the existence of a significant negative correlation between the

strength of the structures and ease of association to the ribosome [174]. Moreover, sequences with no secondary structures associated faster with the 30S ribosomal subunit, and therefore were more likely to form stable initiation complexes, which are determinant for translation efficiency. This effect has been further described by Bentele et al. in 2013, arguing that natural selection has pressured the creation of gene initiation regions with reduced mRNA structures through the use of specific codons to control those regions [20]. Analogous results were obtained by studying the expression of Human IL-10 and Human interferon-α in *Escherichia coli*. Introducing silent mutations to expose the start codon from secondary structures effectivelly improved translation and heterologous expression of both proteins by 10-fold [202]. Similarly, a study showed how the L1 gene from Human Papillomavirus type 16 was modified to avoid the formation of secondary structures when expressed in *Saccharomyces cerevisiae*, again yielding 4-fold higher expression than the wild-type [92].

The concept of minimum free energy is parallel to the formation of secondary structures: the second law of thermodynamics states that the entropy of a system is non-decreasing, and thereby an isolated system tends to equilibrium states where its free energy is minimum. This effect guides the folding of many molecules, including RNA and proteins. In RNA its free energy is bound to the nucleotides without a pair, representing the potential of the molecule to perform work. Thus, minimizing the amount of unpaired nucleotides also minimizes the free energy. The most stable structures are those that yield minimum free energy, and RNA molecules tend to stable conformations. On the other hand, if we are able to reduce the number of pairs, and therefore increase the minimum free energy, we would prevent the deleterious effects described before.

Assessment of RNA secondary structures and minimum free energy (MFE) can be accomplished via numerous algorithms and approaches to structure prediction [98, 111, 176, 205]. The most sophisticated and well known is the fast dynamic-programming approach from Zuker and Stiegler [206], which is based on a first approach from Nussinov et al. [134], and served as a basis for recent methods. Their algorithm attempts to find the structural basepair configuration of an RNA sequence that yields the minimum possible free energy. Implementations of this algorithm can be found in the mFold [205] or Vienna RNA [80] software packages. Other applications focus on performing inverse RNA folding, to produce nucleotide configurations for a given secondary structure, regardless of the gene [6, 14, 30]. However, no method can yet perform the process of obtaining an mRNA sequence that maintains the polypep-

tide primary structure and achieves minimal secondary structure. This gap is likely due to the difficulty in finding the codon-sequence configuration with the highest MFE, requiring calculation of the MFE numerous times, which is an unfeasible task time-wise.

Here we focus on the problem of avoiding stable secondary structures in mRNA molecules by means of maximizing the minimum free energy of the nucleotide sequences, without changing the resulting amino acid sequence. For this, we have developed a method which divides in two: a first part that uses a meta-heuristic approach to explore the space of possible synonymous codon-sequences; and a second part where a fast algorithm calculates a metric that is linearly dependent on the MFE. Thus, the core of our approach resides in computing a pseudo-MFE using a fast method whose results, though not as accurate as current estimators, are highly correlated with MFE. When searching for a synonymous sequence using the meta-heuristic, the pseudo-MFE is used to look for configurations that offer high values of minimum-free-energy, thus effectively optimizing an mRNA sequence.

4.2 METHODS

We consider the problem of having a nucleotide sequence with both coding and non-coding regions, and maximizing the minimum free energy resulting from possible folds by altering nucleotides in the coding region without altering its amino acid sequence. We take advantage of the degeneracy of the genetic code to search for a synonymous gene sequence that maximizes an energy function highly related to the strength of the secondary structure.

The methods are split in four parts: **a)** the search for the best codon combination, **b)** the development of a rapid pseudo-MFE calculation function, **c)** the optimization of that function to maximize its correlation with an accurate MFE measure, and **d)** a linear regression to transform the pseudo-MFE values into more precise bounds.

4.2.1 SYNONYMOUS GENE EXPLORATION

Finding an optimal synonymous sequence is a combinatorial problem which is often impractical to solve in efficient time given the volume of the search space (as shown in previous chapters, 3^N for a sequence with N codons). As a consequence, it becomes attractive to resort to heuristics, such as genetic algorithms and simulated annealing,

that facilitate the exploration of possible sequences, driving the search through regions of the solution space of interest to the problem.

For the MFE maximization problem we used Kirkpatrick's simulated annealing approach [94], which I already showed in the previous chapter to behave quickly and achieve global near-maximum results in codon optimization problems. Thus, starting from the original coding sequence, a number of codons are selected in each iteration to be randomly changed for synonymous ones. The new sequence, including the non-coding regions, is evaluated by the pseudo-energy assessor (described in the following section) which returns a value correlated with the MFE. New sequences with larger values are accepted for the next iteration. However, to avoid local maxima, sequences with lower minimum free energies might also be accepted, according to a probability mimicking the Boltzmann distribution:

$$\exp \frac{e - e'}{k_{max} * 0.9^k}$$

Equation 12 Adaptation of equation 7 in Chapter 3 to the problem at hand. e is the energy value of the current sequence, e' the energy of the new sequence, k the iteration number, k_{max} is the maximum number of iterations and 0.9 is the cooling schedule.The last two parameters were selected after an heuristic assessment to ensure the resulting sequence is near optimal.

The number of codons that are changed in each step also decreases with passing iterations, performing only targeted alterations near the end in order to fine-tune results. The search ends when the maximum number of iterations is reached.

4.2.2 SIMPLISTIC APPROXIMATION TO MFE ESTIMATION

Current tools for secondary structure estimation can accurately measure the minimum free energy that results from the fold of a nucleotide strand. However, such accuracy is achieved by thoroughly analyzing the space of possible secondary structures, and this process may take up to several seconds or even minutes, depending on the size of the sequence. Though they are generally fast enough for a single run, which is the normal use of these tools, they become unfeasibly slow when there is the need for multiple calls, which is the case if one is searching for an optimal configuration of codons and needs to frequently re-evaluate the sequence. For instance, making 1500 calls to RNAfold, in order to evolve an mRNA with 1000 codons (without non-coding

regions), lasts over 6 hours[2].

To overcome this hurdle we have developed an evaluation function that is much faster at analyzing secondary structures, at the expense of less accurate results, and that is highly correlated with the MFE calculated from accurate methods. To reduce the time complexity from the MFE estimation ($O(N^3)$ for RNAfold and MFold, for sequences of N nucleotides) we introduced a simplistic approach with quadratic complexity, which considers all possible single stem-loop conformations and averages their interaction energy, as described in Algorithm 4.1.

The algorithm considers every possible conformation of the mRNA secondary structure using only a single fold (there are approximately $2N$) and, for each conformation, looking for nucleotide pairs that bind (see Figure 4.2). The energy of each fold is the number of hydrogen bonds shared in the interaction regions. The method then returns the average energy for all folds. This approximation does not consider more complex conformations of multiple stem-loop structures or pseudo-knots, which require more intricate formulations that perform a deeper exploration, nor does it intend to yield an accurate energy value. However, it does assume an abstract value representing the minimum free energy that can be obtained from the primary folds of the molecule (pseudo-energy), and that considering all possible single folds further specifies this value to represent a global view of the structure strength. As a result, the algorithm returns a value largely associated with complex MFE predictions.

Figure 4.2 Illustration of the MFE estimation algorithm. All possible folds of a single stem-loop are considered, starting from the 3' end. In each fold, the nucleotides close to the folding region are not considered to interact. The average of the nucleotide-pair contributions of all folds is the result.

4.2.3 FINE-TUNING NUCLEOTIDE INTERACTIONS

To further enhance the statistical dependence of our estimation function with an accurate MFE measure, we chose RNAfold's output as the energy gold standard for

[2]Considering an average of 15 seconds for each call to RNAfold, which is generally a lower bound in a modern personal computer.

Algorithm 4.1 Calculate estimation of MFE

```
function estimateEnergy(seq)
    seqSize ← numNucleotides(seq)
    iBlockSize ← 2                                  ▷ initial block size
    fBlockSize ← seqSize/2                          ▷ final block size
    ls ← 3                                          ▷ minimum loop size
    cEnergy ← 0                                     ▷ Cumulative energy
    for two times do
        b = iBlockSize
        while b < fBlockSize and seqSize ≥ ls + 2 × b do
            b ← b + 1
            subSeq1 ← sequence[0, b]
            subSeq2 ← sequence[ls + b, ls + 2 × b]
            energy = GetEnergy(subSeq1, subSeq2)
            cEnergy = cEnergy + energy
        end while
        sequence = invert(sequence)
    end for
return −cEnergy/seqSize                             ▷ Average energy
end function

function getEnergy(seq1, seq2)
    bondEnergy ← 0
    for i = 0 to numNucleotides(seq1) do
        case (seq1[i], seq2[i]) is
            (G, C) or (C, G) then e ← 3             ▷ Changed to 3.12 after tuning
            (A, U) or (U, A) then e ← 2             ▷ Changed to 1 after tuning
            (G, U) or (U, G) then e ← 2             ▷ Changed to 1 after tuning
            else e ← 0
        end
        bondEnergy ← bondEnergy + e
    end for
end function
```

our function, which offers the current highest performance (0.76 F1-measure) among single-strand secondary structure predictors [111], and the fastest calculation (see

comparison in Appendix B). The tuning was made by changing the contribution of each binding pair (*getEnergy* function in Algorithm 4.1) and assessing the correlation between the approximation function and the results from RNAfold. For that, we randomly selected 48 genes from 6 different species[3], with equal length, since the length of the genes already has a large bias (-97% Pearson correlation) to the minimum free energy.

To perform the optimization we also used a simulated annealing heuristic. Thus, the contribution of GC, AU and GU pairs was changed in each iteration, evaluating their performance by analyzing the correlation between our MFE estimation function and RNAfold's output in the 48 genes. Correlations were measured using Spearman's rank correlation coefficient, which focuses on measuring the extent to which our function increases when RNAfold MFE also increases, using Formula 13.

$$\rho = 1 - \frac{6\sum(f_i - r_i)^2}{n(n^2-1)}$$

Equation 13 Spearman's rank correlation coefficient. f_i is the rank of the value of our approximation function, r_i the rank of RNAfold's output, and n is the number of sequences used (48 genes).

This allows the search algorithm to find binding-pair weights that maximize the dependence to the target output.

Using this method we increased the initial correlation from 0.73 to 0.91, by changing the pair weights from (2, 2, 3) for AU, GU and CG to (1, 1, 3.12). We further confirmed a high linear dependence using Person's product-moment correlation coefficient, which also returned 0.91.

4.2.4 LINEAR REGRESSION

The pseudo-energy values returned by our approach have little physicochemical meaning: they represent only an average over simple folds. However, given the large linear dependence that was created in the previous step, we were able to easily transform the values returned by the MFE estimation function to closely resemble those of an accurate MFE. Though this step is not necessary and does not change the final correlation nor the optimization results, the transformed values become visibly com-

[3] *Aquifex aeolicus*, *Escherichia coli*, *Homo sapiens*, *Mus musculus*, *Rattus norvegicus* and *Drosophila melanogaster*.

parable to those of accurate measures, allowing quick assessment and comparison of MFE values.

Using ordinary least squares we performed a simple linear regression using the energy values of algorithm 4.1 as the input variable, and the MFE given by RNAfold as the observed variable. Furthermore, to better predict MFE, we created two regressions, the first for the wild-type genes, and the second for optimized genes, since they are intrinsically different in their nucleotide constitution:

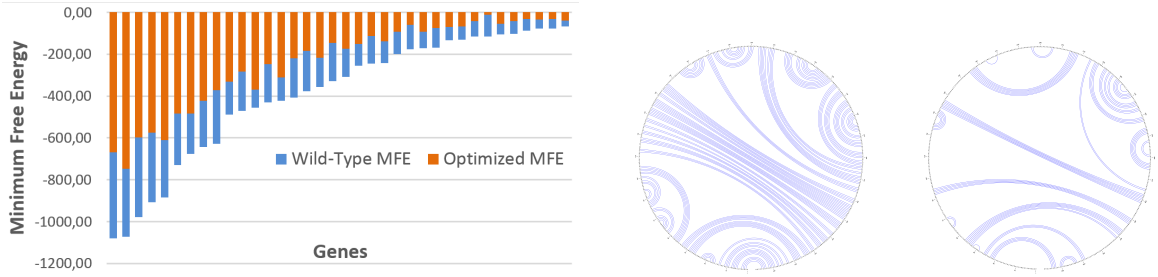
$$\begin{aligned} \text{MFE}_{Orig} &= \text{PseudoEnergy}_{Orig} \times 0.457 - 8.215 \\ \text{MFE}_{Optim} &= \text{PseudoEnergy}_{Optim} \times 0.529 - 15.212 \end{aligned}$$

Equation 14 Regression equations to convert the pseudo-energy into more realistic MFE values. Two equations were created, one for the original gene (Orig) and the other for the optimized gene (Optim), to better fit results.

4.3 RESULTS AND DISCUSSION

To test and evaluate our correlated optimization approach we randomly selected 36 different genes from the same species used in the previous section. By using a different set of genes we avoid biases that might have been generated when tuning the estimation function. To perform an initial assessment we evaluated all genes using both our approximation function and RNAfold, and obtained a Pearson's correlation of 0.99, indicating a perfect statistical dependence between the two approaches. The increase in correlation compared to the results obtained during training is justified by the use of a gene set with random lengths, as opposed to fixed-length, which limited the bias. We proceeded to the evaluation of the optimization of secondary structures by applying our method to the 36 genes and then re-evaluating them with RNAfold to assess evolution (Figure 4.3).

By analyzing the difference between the results of wild-type genes and optimized genes we measured an average 46% increase in the minimum free energy, with a $p\text{-}value < 3 \times 10^{-11}$ (using Student's t-test). As an example, one of the largest evolutions (67%) increased the MFE from -175 to -58 kcal/mol, strongly diminishing the predicted secondary structures. In this optimization, the number of base pairs of the resulting RNA was reduced by 54%, and particularly the number of GC pairs decreased 60%, as depicted in Figures 4.3b and 4.3c.



(a) (b) (c)

Figure 4.3 Secondary structure optimization results. In **(a)** the improvement for each of the 36 genes is shown. In **(b)** and **(c)** the secondary structures of a *Drosophila melanogaster* gene are shown for the original and optimized mRNAs in a circular chart. The circle is created using the gene nucleotide sequence.

One might argue that the amount of guanine and cytosine in genes, which are the nucleotides that produce the strongest pair, could be the main driver behind the optimization process and that the optimization was merely a question of replacing codons for synonymous with less guanine and cytosine. To understand the algorithm bias towards the amount of guanine and citosine in genes we evaluated GC content before and after optimization, measuring an average of 24% decrease, though weakly correlated with the MFE improvement of each gene. We then assessed the role of GC content by adding a rule to our optimization algorithm in order to evolve genes into configurations that simultaneously increase MFE and maintain the same percentage of GC as the wild-type, to ensure that it is codon structure rather than GC content that is controlling the improvement. Results show MFE improvements averaging 28%, without any change in GC content, which represents a 18% decrease in improvement when compared to the original optimization. We also optimized the genes to minimize the amount of GC% and observed improvements in MFE averaging 38%, which is less than our approach. This suggests that the main contributing factor to our algorithm is codon configuration, though GC content also plays a significant role (see Figure 4.4).

Furthermore, to control for GC content and assess its impact in optimization, 36 genes with equal amount of GC (50%) were randomly selected from the same species, and optimized using our approach. Improvements in MFE averaged 43% with a t-test probability of 8×10^{-11}, with results still showing a large correlation between values of RNAfold and our approach (0.99). Also, by adding the previous rule to maintain the same GC% amounts of the wild-type, we measured improvements in

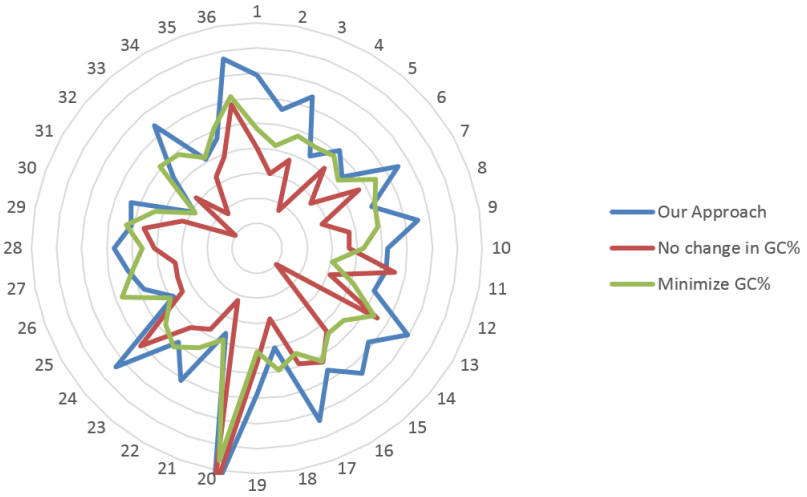


Figure 4.4 MFE optimization controlling GC content. The interior circle represents 0% improvement, while the outer circle 80%. Our approach reaches the best improvements on the majority of genes.

MFE averaging 29% with a *p-value* $< 1 \times 10^{-10}$. Therefore the optimization can be effective despite the original amount or change in GC content.

Considering that our MFE evaluation function is faster but less accurate than RNAfold, we compared our approach with that of optimizing a codon sequence using the same strategy but with RNAfold as the MFE estimation function instead of our correlated method, in order to deduce the loss in accuracy and gain in time. For that, we used the simulated annealing method with the same parameters, replacing only the energy evaluation function by a call to RNAfold. We then performed the optimization in the set of 36 genes (random GC% and length) and collected results for comparison. All final results from both approaches were measured afterwards using RNAfold to build a reliable comparable basis between methods. We found a small gain of 2% when using RNAfold to optimize genes, with more than 35% of the genes having similar or worse results than our approach. Also, higher differences were only found in smaller genes and for genes larger than 250 codons the gain was null, suggesting that as the problem becomes more complex our approach obtains equivalent results to using RNAfold. However, being of quadratic time-complexity, our approach took only a total of 34 minutes to optimize the 36 genes (<1 minute per gene), while using RNAfold took more than 6 days (more than 4 hours per gene)[4]. Parallelizing the

[4]In a computer with Windows Server 2008, Intel Xeon 4 cores 2.67GHz, 4GB RAM

process leads to bounding the problem to the longest optimization, which was around ∼4 minutes using our approach and ∼22 hours using RNAfold (see Figure 4.5). Thus, having no significant loss, and gains in time up to several hundred times, our strategy becomes a feasible approach to optimizing RNA secondary structure.

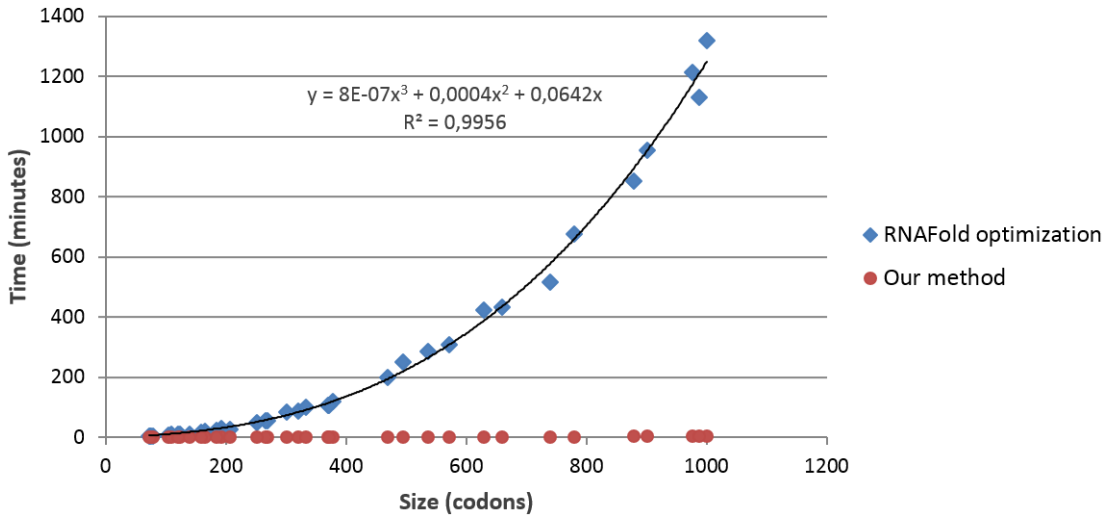


Figure 4.5 Time complexity gain in using pseudo-energy. Using RNAFold to optimize the genes takes a cubic time in relation to the size of the gene (see the regression results), while our method is only quadratic. The loss in accuracy when using our method averages 2%.

We created linear regressions that try to approximate a realistic energy value for our method, and obtained a coefficient of determination R^2 for both equations of 0.99 (n = 36), which reflects the flexibility of the pseudo-energy method in fitting to the energy predicted by RNAfold. Figure 4.6 depicts a comparison of the energy output from both methods in 48 genes from different species.

To promote the use of our strategy we created a command-line tool that uses the pseudo-energy evaluation along with a simulated annealing heuristic to optimize mRNA sequences, available at bioinformatics.ua.pt/software/mrna-optimiser. The tool also allows controlling for guanine and cytosine content. Moreover, by having an evaluation function we easily integrated the method into the Eugene software previously presented, enabling the integrated multi-optimization with other methods.

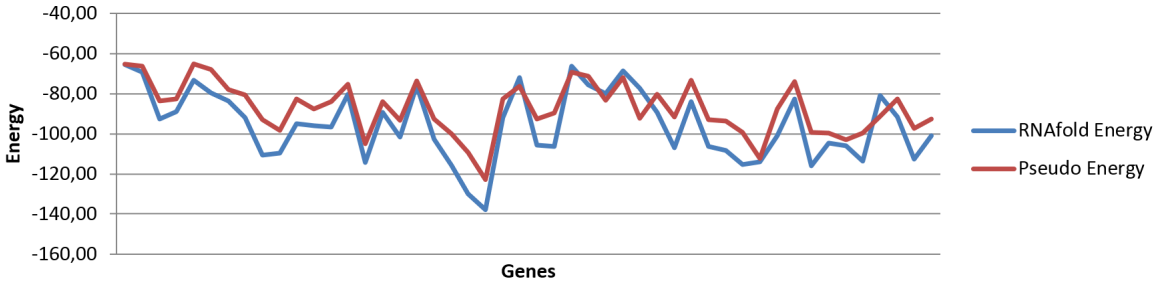


Figure 4.6 Comparison of pseudo and RNAfold energy predictions. 48 original genes (non-optimized) were employed. A correlated landscape is clearly visible.

4.4 CONCLUSIONS

The study of RNA secondary structures is an important area of research in computational molecular biology. Specifically, structure prediction and MFE calculation are prominent subjects in recent RNA literature. Being able to predict the formation of structures has allowed researchers to understand how RNA functions, while reverse RNA folding has allowed building non-coding RNAs that have a specific structure. However, considering the impact of secondary structures on gene translation, a strategy to redesign genes to produce less structured mRNAs that would allow for improved or controlled expression is important.

We presented a first approach to optimizing the secondary structure of mRNA sequences using a fast correlated MFE-estimation method. Though the estimation algorithm was not built for maximum accuracy, results are closely associated with those of using accurate methods, such as RNAfold, allowing for rapid calculation of synonymous genes with improved structures. Overall, our tests indicate an average of over 40% improvement in MFE, as measured by RNAfold.

Our method can be used in combination with other factors that influence gene expression, such those discussed in Chapters 2. By creating a single function able to calculate the pseudo-energy we enabled the use of the techniques explored in Chapter 3 for the aggregation of goals, which is only feasible due to the reduced time complexity.

Besides allowing the optimization of single mRNA sequences alone, the advantage of being fast and accurate (to some extent, as shown) allow performing bulk evaluations, for instance to use in data mining, as seen in the following chapter.

5 | PREDICTING PATHOGENICITY

Due to rapid and large advancements in genetics research and sequencing technology, healthcare is undergoing a paradigm shift that brings the DNA to the center of personalized and preventive medicine. Recent progresses coming from large-scale projects have been able to deploy massive amounts of genetic data from hundreds of humans, providing a comprehensive characterization of the human variome. The gathering of these data has given rise to a new era of human health interpretation through the analysis of the most intrinsic description of an individual: its genome. Thus one of the most important and challenging topics in human genomics research is the interpretation and transformation of such data into informative diagnosing tools.

A substantial amount of work is already being made to understand the impact of genetic variation, which not only dictates individual characteristics but is also the main driver of human pathogenicity. By using the methods discussed in previous chapters to analyze and evaluate genes, we are able to acquire detailed information regarding one of the fundamental steps in the central dogma of molecular biology for any living being: translation. The value of this information in contributing to the analysis of human mutations and consequently pathogenicity through the use of machine learning is studied in this chapter, as we sought to develop an accurate methodology to classify single nucleotide variants according to their pathogenicity. We further develop the field of patient genomic analysis by creating a screening software that detects mutations in patient genes and reports on their pathogenicity using our methodology.

5.1 PATHOGENICITY FEATURES AND LEARNING

5.1.1 HINTS TO PATHOGENICITY

The most common form of genetic variation are single nucleotide variants (SNV), with recent reports from the 1000 Genomes Project revealing an average of 3.6 million SNPs[1] per individual, representing one SNV every kilobase [1, 2]. In comparison, multi-base mutations are 10 times less frequent [126]. Only 24 thousand of these are found in protein coding regions, approximately half of which directly affect the protein (non-synonymous alterations). Moreover, SNVs were soon recognized as the main cause driving genetic variation in humans and an excellent form to map traits [44, 58]. Therefore, efforts are being put in the development of algorithmic strategies to differentiate potentially pathogenic mutations from variants with minor or no consequences. Such algorithms allow screening through thousands of patient variants for pathogenic candidates, performing genetic diagnosis that enable early or preventive treatments. Moreover, by broadening the understanding of factors that impact protein function and play a role in disease, these computational methods may promote personalized medicine.

In chapter 2 we explored several of the most popular approaches in literature to classify genomic mutations according to their pathogenicity or degree to which they might generate instability in a protein. The approaches are based on genetic assessments on the protein at hand, and differ in their classification methodology, with most of them using popular machine learning strategies to perform classification. The assessments are related to genetic or physical-chemical properties, as well as relations of conservancy among species. For instance, as already discussed, the PhD-SNP, Panther and SIFT methods are based on evolutionary conservation, i.e. they account for the level to which the affected region of the protein is maintained in other species with the same protein, as mutations affecting conserved regions might indicate a potential disruption of functional code [33, 129, 180]. Other approaches such as MutPred, PolyPhen2, SNAP and SNP&GO also combine structural and peptide information to analyze conformational and energetic changes that might occur in proteins as a result of amino acid alterations, thereby increasing the ability of the models to infer if the affected region is able to disturb the stability of the protein or organism [3, 28, 161, 186]. Despite the large adoption of homology and protein-based

[1] SNPs (single nucleotide polymorphisms) are SNVs with a frequency of over 1% in the population.

information to create models of pathogenicity, there is overwhelming evidence that mRNA alterations are a major factor causing disease [159].

In fact, amino acid alterations account for only approximately half of the disease-causing variants in the Human Gene Mutation Database [173]. It has been shown that variations in the mRNA, independently of being synonymous [147, 165], have a high impact on the polypeptide biosynthesis and consequently on the protein, mainly due to the additional information carried by the mRNA besides that of decoding [86]. For instance, codon usage frequencies have been reported to play a significant role in translation speed regulation [8, 39, 100], hence variants altering codon usage might disrupt important slow-decoding regions that allow the protein to fold correctly [140]; likewise, changes in codon context, which is the biased use of adjacent codon pairs, have been shown to be able to cause decreased rates of protein translation [41, 124]; variants affecting splicing regulatory sites influence pre-mRNA splicing, with studies reporting that 16% of disease causing mutations affect these sites [123, 189]; changing the local mRNA secondary structure can affect translation initiation and elongation by hampering the ribosome work, and also affecting folding speed [40, 105, 175]; and nucleotide repeats may originate frame-shifting, causing mistranslation and wasted consumption of cell resources [27].

Not withstanding, changes in protein physical-chemical properties, although having a direct impact on the protein, have varying influence on pathogenicity, with a significant percentage of reported amino acid mutations being neutral [2]. Physical-chemical properties include measuring changes in terms of acidity, mass, volume, hydropathy, isoelectric point, net charge and polarity, when an amino acid is replaced in the chain, since amino acids have different chemical compositions (see table 5.1).

	A	C	D	E	F	G	H	I	K	L	M	N	P	Q	R	S	T	V	W	Y
hydropathy	1,8	2,5	-3,5	-3,5	2,8	-0,4	-3,2	4,5	-3,9	3,8	1,9	-3,5	-1,6	-3,5	-4,5	-0,8	-0,7	4,2	-0,9	-1,3
polarity	8,1	5,5	13,0	10,5	5,2	9,0	10,4	5,2	11,3	4,9	5,7	11,6	8,0	12,3	10,5	9,2	8,6	5,9	5,4	6,2
averageMass	89,1	121,2	133,1	147,1	165,2	75,1	155,2	131,2	146,2	131,2	149,2	132,1	115,1	146,1	174,2	105,1	119,1	117,1	204,2	181,2
isoElectricPt	6,0	5,1	2,9	3,2	5,5	6,1	7,6	6,1	9,6	6,0	5,7	5,4	6,3	5,7	10,8	5,7	5,6	6,0	5,9	5,6
vanDerWaals	67,0	86,0	91,0	109,0	135,0	48,0	118,0	124,0	135,0	124,0	124,0	96,0	90,0	114,0	148,0	73,0	93,0	105,0	163,0	141,0
netCharge	0,0	0,0	0,0	0,0	0,0	0,2	0,0	0,0	0,0	0,1	0,0	0,0	0,2	0,0	0,0	0,0	0,0	0,1	0,0	0,0
pK1	2,4	1,9	2,0	2,1	2,2	2,4	1,8	2,3	2,2	2,3	2,1	2,1	2,0	2,2	1,8	2,2	2,1	2,4	2,5	2,2
pK2	9,9	10,7	9,9	9,5	9,3	9,8	9,3	9,8	9,1	9,7	9,3	8,7	10,6	9,1	9,0	9,2	0,0	9,7	9,4	9,2
pKa	0,0	8,2	3,9	4,1	0,0	0,0	6,0	0,0	10,5	0,0	0,0	5,4	0,0	0,0	12,5	5,7	5,5	0,0	5,9	10,5

Table 5.1 Amino acid physical-chemical properties. Values for each amino acid are shown for the most important properties [23, 56, 106]. The table is colored according to the scale of each property.

Significantly changing some of these properties might cause a diminished or even absence of functionality in the resulting protein, depending on other factors such as the size of the protein, the importance of the affected region, and the impact of that property in the polypeptide folding. For example, the presence of cysteine amino acids in proteins can induce the formation of disulfide bonds, which make the protein structure much more stable. Therefore these bonds are usually found in proteins that function in harsh environments, such as digestive enzymes like *pepsin*, or proteins that are too small and require additional help to hold a stable conformation, such as insulin. Replacing a cysteine amino acid can therefore cause structural and stability changes, rendering the protein less resistant to denaturation[2]. Although these are arguably the most recognized and influential properties, the Amino acid Physical-chemical property Database (APDBase) holds almost 250 redundant properties that have been independently measured [117].

Moreover, affecting properties such as the hydropathy can also have a large impact in folding, since the main driver of protein folding is the water-solubility of amino acids. Highly hydrophobic amino acids tend to avoid the contact with the water environment by clustering in the hydrophobic core of the protein, which becomes surrounded by the remaining proteins and leaving the outside of the protein formed by hydrophilic amino acids. Significantly altering the hydrophobicity of a region that was intended to form the core of the protein can impede its correct folding and thereby rendering it nonfunctional.

Also, proteins have different degrees of importance, relating to their role in the organism: affecting central proteins is far more deleterious than affecting proteins with less important roles, or with compensatory or redundant systems. In that matter, a few initiatives have been developed to annotate genes and proteins according to their function and interrelations. The Gene Ontology (GO) is one of those efforts, which tries to unify and create a universally accepted and consistent terminology to describe gene products in three domains related with their biological roles:

- molecular function - describes the biochemical activity of the gene or gene product. Broad examples are “enzime” or “transporter”.
- biological process - related to the purpose or overall goal of the gene or gene

[2]Denaturation occurs when a protein loses its natural conformation, usually in the presence of external agents such as strong pH, solvents, salts or heat. This results in the loss of activity by that protein.

product. It is often associated with chemical or physical transformations. Examples are "cell growth" or "signal transduction".

- cellular components - refers to places inside the cell, or parts of it. Examples are "ribosome" or "nuclear membrane".

Using an ontology allows the creation of a semantic representation of the relations between the factors involved with gene products. The ontology can be graphically represented as a graph involving all biological components and their relations, as depicted in Figure5.1, where a gene product can be annotated with one or several of these terms. Concepts are represented as nodes of the graph, and relationships are described in the edges that connect two nodes.

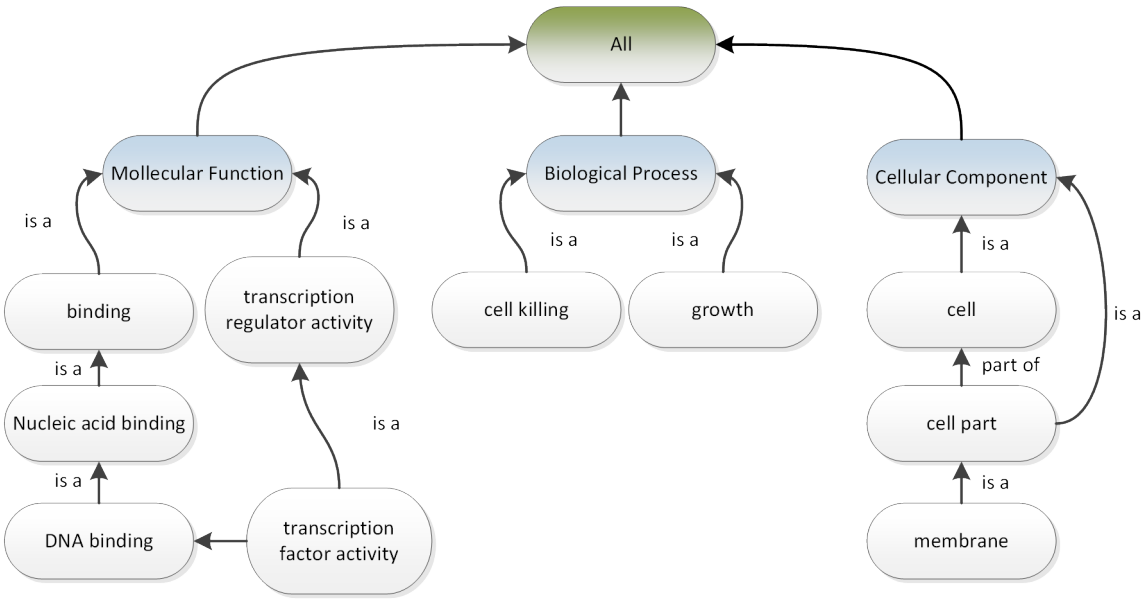


Figure 5.1 A Gene Ontology section. Two types of relations can be seen, the "is a" and the "part of". The first indicates a subclass-class relation and the second a part-whole relation. (image adapted from the work of Pesquita *el al.* [142])

By also entailing the relationships between and among roles and components, the GO enables searching biological databases by function and associations, and provides a comprehensive source of knowledge to be used when creating models of living systems. This last advantage is essential to understanding how a mutation affecting a protein might also affect the organism by an intertwined succession of interactions with other actors and processes in the surrounding biological system. An example of

a tool resorting to GO is SNPs&GO, which provides a service to predict if amino acid alterations are deleterious by using functional annotations coming from GO [34].

Overall, approaches in literature are still relying on homology and conservation as the backbone for the analysis of mutations. Though, in principle, homology does reveal the importance of the affected region as a measure of how much the zone was kept among species throughout evolution, it lacks essential ingredients to measure the impact in proteins or even pathogenicity. For instance, factors involved in the translation step are largely ignored in literature. In 2001, Wang and Moult mapped mutations known to cause disease onto their corresponding protein structures and found that only 83% of the mutations affected protein stability[192]. That means the remaining mutations cause other factors, such as affecting the translation and reducing the amount of expressed protein. The use of curated knowledge is also rarely adopted, despite the fact that it could potentially indicate the importance of a protein inside an organism. The general disposal of such important clues to pathogenicity justifies the still relatively low accuracies in current prediction systems. Even in aggregation strategies such as PON-P, which brings together several approaches (SIFT, PolyPhen 2, SNAP and PhD-SNP), the lack of more informed features leads to a score of 80% accuracy [137], which is only fractionally larger than the individual scores.

5.1.2 MACHINE LEARNING

To be able to perform computational predictions on whether a variant is pathogenic or neutral to its carrier, an algorithm must be able to distinguish variants based on given hints. Hints might be evaluations, retrieved data, or assumptions, and the goal of machine learning is to generate an algorithm to represent and generalize those hints such that new unseen samples can be correctly categorized. For example, one of the most widely used systems based on machine learning is the detection of spam in emails. Using the contents and meta-data as the hints of an incoming email, the system must classify the email as spam or normal.

Therefore, the core hypothesis of machine learning lies in that learning to generalize a given input data in terms of building a classification structure will allow inferring the classification of novel input data by mapping it to the constructed structure. It intimately assumes that the initial input data is largely representative of future data, that is, the structure of future data in somehow known, and therefore the input data plays a crucial role (arguably the most important) in the ability of a machine learning model to generalize.

The learning procedure varies wildly depending on the generalization methodology. Perhaps the most classical approaches can be divided in two classes: supervised and unsupervised learning. The former assumes the availability of a prior classification additional to the input data, where the learning procedure tries to fit the hints, examples of which are Artificial Neural Networks, Support Vector Machines and Decision Trees. The latter does not require a prior classification but rather tries to find the underlying structure of the input data regardless of its class, for instance by clustering data points. Examples of unsupervised learning include K-means, Hidden Markov Models and strategies of dimensionality reduction such as Principal Component Analysis and Singular Value Decomposition. However, most prediction systems rely on supervised learning due to being more suitable to classification tasks, often yielding better performances, and allow learning by example.

For example, in order to classify data, Support Vector Machines try to find the maximal margin that separates instances by class in a geometric space, and output the separating hyperplane at the center of the margin [46]. That separation is found by assuming that each hint represents a different dimension in a high-dimensional space, and that data instances represent points in that space (Figure 5.2a). The hyper-plane that better separates instances of one class from the other is the best generalization. New data is classified by determining on which side of the hyperplane it belongs and hence to which class it should be assigned. However, some input spaces might not be linearly separable, and therefore kernel functions are used to map the input space into a higher-dimensional feature space where the data can be more easily separated (Figure 5.2b). The optimal kernel to use depends on the classification problem and the available data, and usually has specific parameters which can be controlled to fine-tune the SVM performance.

A well-designed machine learning experiment requires several steps, such as the extraction of data, the creation of the model and its evaluation, and must consider several major issues related with generalization, such as the trade-off between bias and variance (bias-variance dilemma), the dimensionality of the input, or redundancy and dependency in data. These are explored as follows.

Learning steps

Regardless of the supervised learning method, a necessary first step towards creating an abstract model capable of fitting data is extracting data itself and assembling a dataset composed of the collected samples, their features (hints) and annotated or

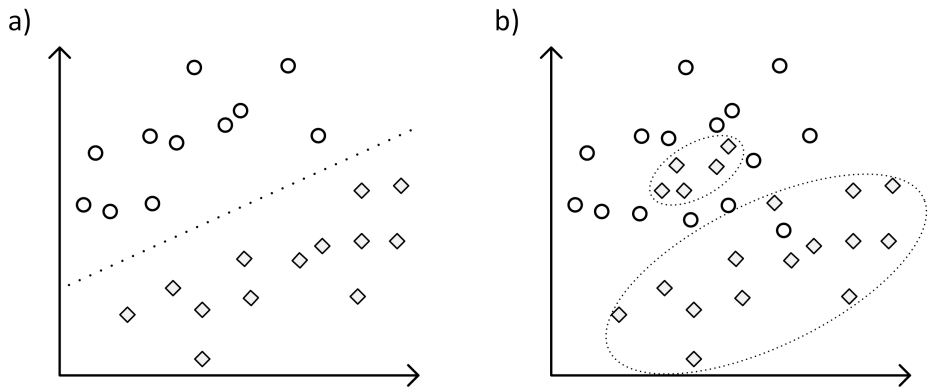


Figure 5.2 Example of Support Vector Machines. **a)** Instances of data are plotted in the chart using two abstract hints (used in the horizontal and vertical axis). Circles and rhombi distinguish their classification. Support Vector Machines try to find the line that optimally separates the classes. **b)** A linear separation is not possible because many instances would be misclassified. Using kernels allows mapping the input space into a higher dimension that allows separating the classes with less error. In the image, only one example is misclassified after using a Radial Basis Function (RBF) kernel.

observed classification. This first step ensures that a workable input with vector shape can be used by the learning algorithm to map the output as a function of the input. In our theme, that represents collecting a set of annotated variants, i.e. for which we know the pathogenic outcome, and evaluate each of them according to several hints (from now on, features).

A second step involves pre-processing the gathered data. This is a more generic step that is highly specific to the case at hand (Figure 5.3a). To ensure the quality of the input data before the actual learning, one must deal with inconsistent data, missing values, unscaled features, noise, outliers and other issues that might compromise the learning algorithm or the generalization capabilities of the learned model. Common pre-processing approaches include:

- Removing instances that have inconsistent or empty data, or filling the invalid features using sampling from similar distributions;
- Selecting the instances that represent a good set of examples to the learning phase, for instance by representing real-case examples;
- Scaling in order to keep all features inside the same boundaries and avoid bias towards features with larger values. This is often achieved by reducing all features to a null average and unity standard deviation;

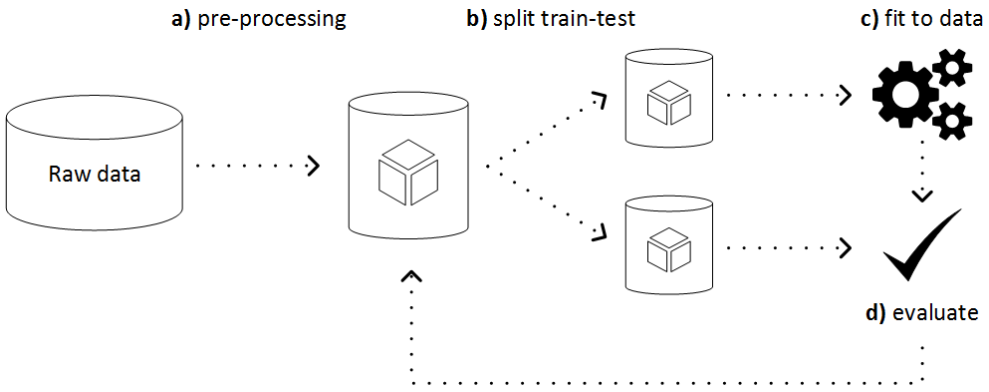


Figure 5.3 General supervised machine-learning steps. After collecting data and pre-processing it, the train-test-evaluate cycle beings, starting by splitting the data set into two parts, then using one of them to fit a model, and finally evaluating the model in the unused data. To correctly measure the generalization power of the created model, the process is repeated several times using different data splits, reducing the data bias.

- Performing feature selection to reduce the dimensionality of the input space and remove noise from data. This approach is largely studied, and there are many strategies to it including removing features that are redundant, share little information with the class (i.e. are uninformative), or that were proved to be less useful for the learning method;
- Normalizing instances such that all instances have an equal vector magnitude, avoiding bias towards instances with larger or smaller magnitudes. This is usually performed by reducing the instance vector to a unity magnitude;

Having the dataset prepared, the learning cycle may begin. The cycle is generally made of three steps: the dataset is split into N equal parts (Figure 5.3b), using $N-1$ parts to perform the actual fitting of a model (training, Figure 5.3c), and the remaining part is used to evaluate the created model in unseen data (testing, Figure 5.3d). This is called cross-validation, and the process is repeated $N-1$ times, excluding a different part to use for evaluation in each iteration. At maximum, N can be the number of samples in the dataset, meaning that the model is fit to every sample except one, which is used to test the model, and the process repeated for every sample. This is called leave-one-out cross-validation. Cross-validation is a widely used approach because it confers the learning cycle with a scientific approach to correctly evaluate the generalization capabilities of the created model, as the testing data is unknown by the learning method in the training phase and therefore its evaluation

must be similar to that of any new unseen data. Again, this assumes that the dataset prior to splitting is representative of the real-world distribution of data. Also, cross-validation ensures the model is trained and tested using every part of the dataset, thus making evident the presence of any bias in the data.

It should be noted the importance of performing some of the pre-processing approaches inside the learning cycle instead of prior to it. The reasoning behind that is that some approaches, such as scaling and feature selection, operate over all instances (and even the class) by analyzing them and then performing transformations based on that analysis. The consequence of those operations is that information is shared between instances, or even between the class and features, effectively introducing bias and prior information into the data.

The evaluation of the model can be attained using numerous approaches. The goal is to verify the extent to which the created model is generalized and not overfit to the dataset. Four base concepts can be easily outlined: the instances that were correctly classified by the model as belonging to the positive class are called the true positives (TP); similarly, true negatives (TN) are negative class instances that were classified as negative; the errors of the model stand on the instances that were incorrectly classified as positive (false positives, FP) or incorrectly classified as negatives (false negatives, FN). Combinining these values leads to several well-known formulations that serve as the basic evaluation of machine learning systems. One of the most popular and important measures is accuracy, which is the ratio of true classifications (TP+TN) in relation to the amount of classified instances:

$$Accuracy = \frac{TP + TN}{TP + TN + FN + FP} \tag{5.1}$$

Accuracy is perhaps the most used metric of evaluation, but bears the issue of being biased if the amount of positive and negative samples in the dataset is not balanced. Another two important methods are precision, which is the proportion of positive samples among those that were classified as positive: $precision = \frac{TP}{TP+FP}$; and recall (or sensitivity), which is the proportion of positive samples that were correctly classified, calculated with $recall = \frac{TP}{TP+FN}$ [114]. An assessment metric called F-measure contemplates both precision and recall, by performing an harmonic mean of

both:

$$F_\beta = (1+\beta^2)\frac{precision \times recall}{precision + recall} \tag{5.2}$$

The most common value for β is 1, granting equal weight to precision and recall (F1-measure). This method is more effective than accuracy, as it can be used in unbalanced settings. However, as pointed out by Powers in 2011, reducing the equation reveals that the F-measure completely ignores the amount of true negatives, an important factor when evaluating a classifier [149]. Employing the correlation between the true and predicted classifications is another strategy, and one form of measuring it is through the Matthews Correlation Coefficient (MCC) [118]:

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}} \tag{5.3}$$

The MCC can be translated as being the geometric mean of markedness and informedness, two other important measuring methods [149]: $informedness = (recall - \frac{FP}{FP+TN})$ and $markedness = (precision - \frac{FN}{TP+FN})$. Lastly, but not less important, are Receiver Operating Characteristics (ROC) curves [57]. ROC analysis plots the true positive rate (recall) against the false positive rate (also known as fall-out, $\frac{FP}{TP+FN}$), as the threshold that generates the classification decision is varied. For instance, Support Vector Machines use 0 as the threshold to decide whether an instance is from one class or another. Varying the threshold will change the recall and fall-out, allowing to assess the best thresholds, and the generalization capabilities of the learning method. The ROC curve that is created will ideally be closer to the (0,1) corner where the false positive rate is minimal and the true positive rate is maximal. Therefore, the area under the ROC curve (AUROC) is often also used as a measure of classifier performance.

LEARNING ISSUES

The first issue with machine learning, as well as with optimization, is that according to the no-free-lunch theorem there is not one single method that performs better than all others [198]. In fact, the theorem showed that for any approach, its average performance over all problems will return the same as any other. That means that

the choice of method is highly dependent on the problem at hand, hence suggesting the need to look for a good candidate approach before proceeding with further work.

Another issue to take into consideration is the bias-variance dilemma. This problem relates to the trade-off that often exists between the ability of the model to generalize to new data and its ability to capture the structure inherent to the training data [70]. Ideally a learned model has no bias, that is, it has optimal performance in all the data used to train it, but also has no variance, meaning that unseen data settings (unseen values, unseen combinations, noise, etc.) also behave optimally, which translates into optimal generalization. However, this is rarely the case, as creating a model with zero bias can be very hard depending on the dataset at hand, but even when creating such model it would hardly generalize to any other data, due to be over-fitting the training data. Therefore, it is common that problems are bounded by a trade-off between bias and variance, and finding the right balance is important in order to guarantee the practicability of the created model. Nonetheless, to overcome this issue cross-validation techniques can be used since they maintain a separate testing set of data instances allowing an unbiased performance evaluation, and the training-testing cycle enables a formal assessment of variance.

Another strategy to tackle the bias-variance dilemma is employing techniques of dimensionality reduction. The high dimensionality of the input space is another important issue in machine learning. It is intimately related with geometrical spaces and the fact that using more dimensions leads to having a sparser representation of data. This occurs mainly due to most machine learning algorithms resorting to L^p spaces, e.g. the Euclidean space, to perform distance measurements. As seen in chapter 3, the Minkowski formulation (equation 9) depends on the dimension p of the space where the measurement is made, and as p grows the equation becomes unable to efficiently discern the distance between different pairs of instances, as points in space tend to be equidistant. In fact, in high dimensional spaces, most of the volume is concentrated near the boundaries of the space, and this effect increases exponentially with the number of dimensions. For instance, in Figure 5.4 a small example is shown where each object has a smaller version (shown in red) that represents 1% of the size/volume of the larger object. In one dimension, the red line has a length of exactly 1% the length of the black line. In two dimensions, in order for the red square to have 1% the area of the black square, its edges have a length of 10% of the edges of the black square. In three dimensions the same happens, and the red edges now have 21% of the edges of the larger cube in order for its volume to

represent 1% of the volume of the involving cube. In 1000 dimensions, the edges of the red hypercube would have $0.01^{\frac{1}{1000}}$, which is approximately 99% of the edges of the outer cube, but the red cube would still have only 1% of the size of the black cube. That means that in high dimensions, the majority of the volume is near the edges, and therefore measuring Minkowski distances becomes less efficient. Several machine learning models rely on Minkowski or derived distances, such as Support Vector Machines. Another issue with high dimensionality is that having more dimensions exponentially increases the amount of data necessary to have a good sample of the possible range of combinations between variables. For instance, when adding two dimensions that can have 10 discrete values each, the number of value combinations that must be considered grows by $10^2 = 100$ fold. This effect is known as the *curse of dimensionality*.

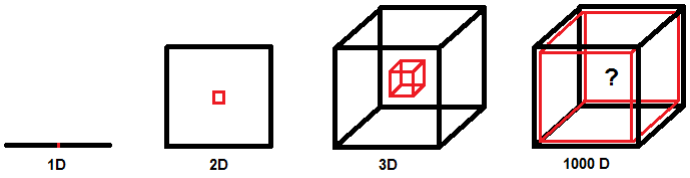


Figure 5.4 Illustration of the dimensionality curse. The red versions represents 1% of the size/volume of the black versions. As the number of dimensions grows, the red version becomes closer and closer to the edges of the black version. For instance, in an hypothetical hypercube of 1000 dimensions the inner red cube's edges would be 99% of the edges of the outer cube, while still being only 1% of the volume.

To tackle this issue the number of dimensions should be reduced, for instance using feature selection to find the best minimum combination of dimensions (by removing irrelevant or low-gain dimensions), or matrix decomposition strategies such as principal component analysis or singular value decomposition, which allow selecting the most informative dimensions. Dimensionality reduction approaches also tackle another important issue, which is that of redundancy and dependence in data, which introduce noise and inconsistencies in the dataset, hampering the learning by some methods.

5.2 METHODS

5.2.1 FEATURE EXTRACTION

To obtain annotated variation data to use as our input data we gathered a total of 56802 unique single nucleotide variants from the coding regions of 13779 gene transcripts. The mutations were extracted from several Locus Specific Databases (8 databases, 8501 variants, 85% classified as pathogenic or likely pathogenic) [60], SNPdb (48307 variants, 45% pathogenic, with neutral polymorphisms retrieved as those having a global minor allele frequency above 20% in the Phase I 1000 Genomes Project population) and Cafe Variome (691 variants, all pathogenic), as detailed in Table 5.2.

Locus Specific Databases	# Variants
chromium.liacs.nl	2569
chromium.liacs.nl - colon	4009
databases.lovd.nl	985
dmd.nl	8292
dmd.nl - eye	1840
eds.gene.le.ac.uk	466
grenada.lumc.nl	5464
lovd.i-med.ac.at	75
TOTAL *(coding SNVs only)*	**8501**

dbSNP	# Variants
ncbi.nlm.nih.gov/SNP/	48307
TOTAL *(coding SNVs only)*	**48301**

Café Variome	# Variants
cafevariome.org	1368
TOTAL *(coding SNVs only)*	**691**

FINAL Datasets	# Variants
all unique valid coding SNVs	56802
balanced (half neutral)	**55550**

Table 5.2 Source of our dataset variants.

It is important to note that neutral mutations from SNPdb were retrieved as those that were very frequent among the population, hence assuming that when a variant is present in more than 20% of the population it must not constitute a pathogenic variant. Other approaches have assumed lower values, such as 5% and 10%, but given the amount of variants now available we sought to fabricate a more consistent dataset. We confirmed the mutations against their mRNA transcripts, and only valid single nucleotide variants in coding regions were selected. We then created a balanced dataset (50% pathogenic, 50% neutral variants) with 55550 random variants to perform experiments, measurements, and create a prediction model. About 28% of the variants are synonymous (silent) alterations.

To extract variant, gene and protein characteristics (i.e. the features to be used by the machine learning methods) we employed the methods explored in the previous

chapters, along with some manually created algorithms for specific characteristics, generating 152 features (see Appendix D).

For gene features, we calculate several characteristics mentioned in literature as having the potential to impact translation or the overall protein biosynthesis process, such as codon usage (relative synonymous codon usage), codon context (codon pair bias), out-of-frame stop codons, nucleotide repeats, mRNA secondary structure energy (using the method described in Chapter 4), variant positioning (in the codon and in the gene), GC content, proximity to splicing sites, presence of exon splicing regulatory sequences, potential to create nonsense alterations, the original and mutated nucleotides, and several variations of these. We used the human genome (Genome Reference Consortium GRCh38) to pre-calculate Relative Synonymous Codon Usages and Codon Pair Scores for all codons and codon pairs. Hidden stop codons were calculated by looking for UGA, UAA and UAG codons in all possible out-of-frame codons surrounding the mutated nucleotide. Nucleotide repeats were calculated as the number of consecutive repeated nucleotides (3 or more) within a window of seven codons. Splicing features were two fold: the first was created by retrieving the exon positions for the affected transcript to calculate the distance to the nearest splicing site; the following features used two lists of exon splicing enhancers and silencers [191] to calculate a score that represents the amount of possible matches between the affected region and a known exon splicing regulation sequence. The features represented how much the score changed between the normal and mutated versions. The score was obtained by summing the Levenshtein distances[3] between the affected region and every splicing regulatory sequence on our lists, effectively indicating how much the region encloses a regulatory sequence.

Whenever necessary we transformed the values of features using a natural logarithm to ensure a better distribution of values if they encompassed large differences in magnitude.

For protein characteristics, we describe alterations in the chemical properties of the amino acid sequence: hydropathy, polarity, mass, volume, acidity, net charge, and isoelectric point, as well as structural changes by predicting the protein secondary structure using PsiPred [89, 120] and evaluating changes in the affected region (number of β-strands, α-helices and coils in the affected region). Variations of these features were also extracted, such as calculating gene-wide and windowed (7 codon-

[3]The Levenshtein distance measures the difference between two strings in terms of the number of single-characters changes required to transform one string into the other.

s/amino acids) properties. We also use data from a published database of expected Ramachandran plots to assess the stereo-chemical effects of amino acid replacements, which reflect important structural alterations in the protein [182]. For instance, when a mutation caused an amino acid alteration, we retrieved what the expected PhiΦ-PsiΨ angles (Figure 5.5a) should be between the affected amino acid and the previous and next amino acids, and calculated the difference to the expected dihedral angles of the wild-type protein sequence (Figure 5.5b). We did this by creating an algorithm that searches the database for a pair of consecutive amino acids, and assesses what are the most likely Phi-Psi angles between them, and then calculating the difference between the native and mutated angles.

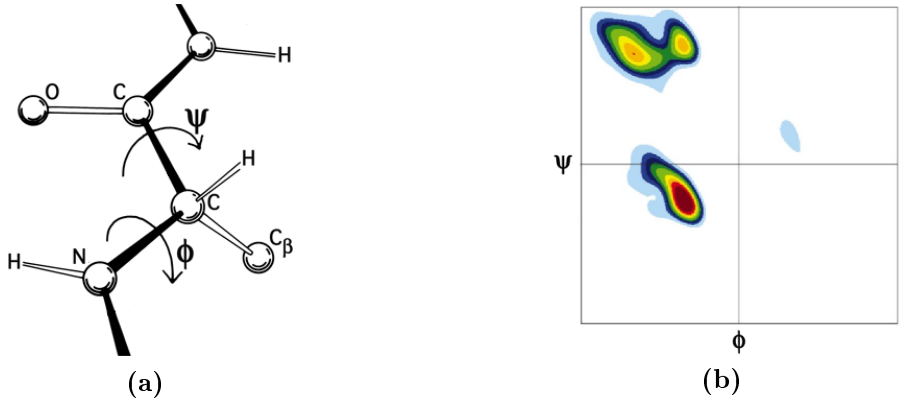


Figure 5.5 Dihedral angles and Ramachandran plots. In **a)**, a section of the backbone of a protein is shown, and the Phi (Φ)-Psi (Ψ) angles are displayed next to the central Carbon atom. Ramachandran found that depending on the neighboring amino acids, the Phi-Psi dihedral angles assume very limited values, as seen in **b)**,which shows the most common combinations of angles for Alanine (blue the least common, red is the most common).

Furthermore, to categorize proteins according to their role we explored the structure of Gene Ontology (GO) annotations. Since there are thousands of GO terms that can annotate proteins, we wanted to avoid having every term as a feature of our dataset to prevent large dimensional spaces and bypass the problems associated with it. Therefore, for each protein to which the variants in our dataset belonged, we evaluated all its annotated GO terms and looked instead for their correspondence in the second level of the gene ontology tree, rendering a limited and fixed set of annotation. This was performed by using a protein-to-GO mapping supplied by UniProt which allowed us to find which GO terms were manually attributed (curation) to a given protein. We then traversed the Gene Ontology graph, which is provided as a entity-relation-entity list, finding for the ancestor entities that belong to the second

level of depth of the graph, as illustrated in Figure5.6.

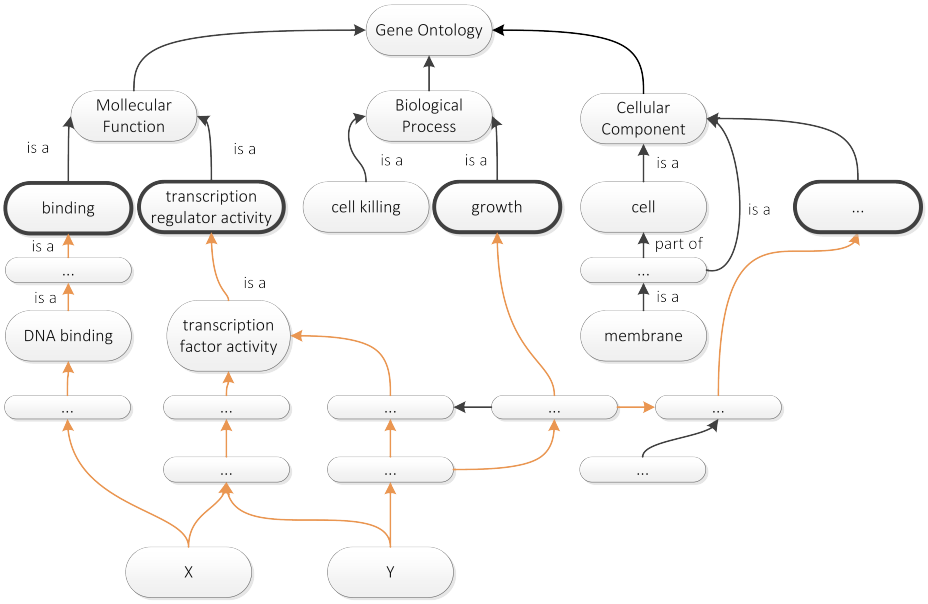


Figure 5.6 Gene Ontology traversing algorithm illustration. A protein annotated with the GO terms X and Y (at the bottom) will be asserted as positive features for several terms of the second level of Gene Ontology (third line of the graph). This is performed by traversing the graph upwards, for each parent term, until the second level is reached.

There are 61 terms in the second level, but four of them had no correspondence in the proteins of our datasets: GO:0019012 (virion), GO:0055044 (plant symplast), GO:0036370 (D-alanyl carrier activity), and GO:0097423 (mitochondrion-associated adherens complex). The first two are external to the expected human GO terms, and the last two were not found due to the lack of proteins/diseases associated with those terms in our dataset. The complete set of Gene Ontology terms that were used as features in our dataset is available in Appendix C.

As an example, a variant whose affected protein is annotated as a basement membrane (GO:0005604) would be characterized as being related to extracellular region (GO:0005576) and extracellular matrix (GO:0031012), both terms from the second level of the gene ontology that are attained by climbing up the graph.

5.2.2 CLASSIFICATION

Using the 152 characteristics extracted from the 55 thousand reported variants, we sought to build a classification model able to predict if a coding single nucleotide

variant has a pathogenic outcome or not. To perform the prediction we tested 17 machine-learning classifiers in order to find the classifier that best matches our prediction needs: Decision Trees, Random Forests, Extremely Randomized Trees, Perceptron, Neighbor-based classifiers, Naive Bayes, Support Vector Machines (SVM), Ridge Classifier, Gradient Boosting, and several variations of these. The complete list is in Table 5.3.

Type	Variation	Type	Variation
SVM	Linear	Naive Bayes	Gaussian
	Polynomial		Bernoulli
	RBF		Multinomial
	Sigmoid	Neighbors	kNN
Neural Networks	Perceptron		Nearest Centroid
	Bernoulli RBM		Radius
Decision Trees	Decision Tree	Other	Gradient Boosting
	Random Forest		Ridge Classifier
	Extra Random Trees		

Table 5.3 List of tested classifiers. Besides the variations within each type, several parameters were tested for each classifier. For example, we tested the polynomial kernel of SVMs using 2 to 6 degrees.

All classifiers were tested in experiments by performing a 5-fold cross-validation (data divided in 5 parts, training in 4 and testing in 1, and then rotating). Final tests used a 20 fold cross-validation instead, to assess a more real predictive power. We also tested several pre-processing strategies: scaling, normalizing, binarizing (transforming feature values into 0s and 1s) and several combinations of these. It is important to note that to avoid information leakage between instances of our dataset, and between the training and testing sets, the pre-processing was included within the cross-validation step. Hence, upon training the train data was pre-processed, and a pre-processing model was saved, which was used later to pre-process the test data. We performed the machine learning and pre-processing computations resorting to Scikit-learn, a Python library for machine learning and data mining [141].

In order to assess the statistical significance of our model's ability to fit the dataset and find a real data structure, besides cross-validation we also employed a randomization test described by Ojala and Garriga [136]. In their method, the class of our samples (pathogenic/neutral) are randomly permuted to estimate a null hypothesis of the dataset. Then the fraction of permutations that yielded results better than the original labels did is inferred. This strategy enables evaluating how likely the ob-

served accuracy of our models would be obtained by chance, and obtaining a p-value representative of our model's accuracy.

We also used several techniques to perform feature selection and reduce the dimensionality of our dataset. In terms of matrix decomposition, we used principal component analysis, fast independent component analysis [24], and latent dirichlet allocation [25] to verify the minimum amount of components we could achieve without loosing performance. We employed a correlation matrix using Pearson's correlation to create distance maps and avoid features that were mostly redundant.

Evaluating features and their importance to the classifier brings understanding on the biological impact of each characteristic. Therefore, we used several techniques to evaluate the importance of each feature and draw conclusions:

- Information gain, or Kullback-Leibler divergence, compares the distribution of a feature with the distribution of the class. It measures the amount of information that is lost when trying to approximate the class using the feature.
- Recursive Feature Elimination, iteratively uses a machine learning model to perform training-testing and removing the feature that contributed the least to the learning method in each iteration. This is performed by using the weights output by the classifier for each feature, and thus only works with some methods. At the end, features are ranked according to the number of iterations they lasted. The last remaining feature is the most important.
- Weights when using classifiers, for instance employing extremely randomized trees allows extracting the importance of each feature after the method has learned. The same can be achieved using linear-kernel support vector machines.

5.3 RESULTS AND DISCUSSION

5.3.1 DATA ANALYSIS

We analyzed the retrieved variants and the 152 calculated features, and found several patterns in the data, specially patterns that indicate tendencies for pathogenic mutations to occur under certain circumstances. These not only explain the predictive power of the features under machine learning settings, but also yield important insights into biological factors related to disease etiology, and have the potential to

expand the knowledge regarding the influence of genetic mutations in humans. We explore these findings in the next paragraphs.

By analyzing characteristics related with the mRNA but not directly with the protein, we found clear discriminatory patterns:

- The position of the variant within the codon was biased in distribution of pathogenicity, which relates to the degeneracy of the genetic code[4], with variants occurring in the third position having an 86% chance of being neutral (n = 18348), and other positions a 68% chance each of being pathogenic (n = 37202);
- The presence of nonsense alterations (i.e. alterations that lead into creating a stop codon, which occurred in 11% of our dataset) also led to a large but expected 98% chance that the variant is pathogenic. The remaining 2% are explained by variants occurring at the final region of the gene where the insertion of a stop codon has a lower impact in the protein conformation;
- Changes in codon usage, with values below -0.5 have an 83% chance that a variant is neutral (n = 4543). Also, generally, there is a strong tendency for increasingly negative changes resulting in more neutral variants (Figure 5.7a);
- Gene size, with large genes having a lower probability of causing pathogenicity when affected by a mutation (88% of mutations in genes larger than 5200 nucleotides are neutral, n = 864), likely due to the low impact a single amino acid has in the overall structure; In a related observation, the position of the variant is also significative, where variants occurring at the beginning of the gene tend to be increasingly pathogenic, whether variants in late positions tend to be increasingly neutral (see Figure 5.7c).
- Changes in codon GC content, where variants causing the loss of a guanine or cytosine nucleotide have 64% probability of being neutral (n = 16644);
- Changes in codon context (as measured by codon pair score), where variants with increasing changes, both positive and negative, tend on being neutral (Figure 5.7b). This result is counter-intuitive, as it indicates that changes in a gene property tend on being neutral, and not performing changes tends on being pathogenic;

[4]Changing the last nucleotide of a codon frequently yields a synonymous codon, hence representing a silent mutation.

- Changes in local mRNA secondary structure energy (7 codons window), as measured by our algorithm, with variants causing an increase in energy being 18% more likely to be pathogenic than the ones causing negative changes;
- The distance of the mutation to the nearest edge of the exon, where variants farther away from exon edges have a higher chance of being neutral, up to a maximum of 80%.

These patterns became more visible when calculating the distributions of each feature, and grouping the result by pathogenicity, as depicted in Figure 5.7. We further calculated the ratio of pathogenic-to-neutral mutations for the whole variation of each feature, using a moving window. The ratio enables a localized evaluation of each feature by assessing the probability of finding a pathogenic variant for any value of a feature. This probability is represented in Figure 5.7 by the black line, which was smoothed to reduce noise using a LOWESS (locally weighted smoothed scatterplot) regression.

As expected, we also found that features describing amino acid changes were highly descriptive of protein dysfunction and caused a large impact in pathogenicity when altered. For instance, we observed a 98% chance of a variant being pathogenic in large mass changes (local mass difference larger than 12 Da, n = 5943), 94% in large volume changes (local Van Der Waals volume difference larger than 10Å^3, n = 7271), and 91% when polarity is significantly altered (changes larger than 4.9, n = 10142, see Figure 5.7d).

Furthermore, when an amino acid is replaced but side-chain chemical properties remain unchanged, we found a significant chance that the alteration is neutral. For example, the non-alteration of any of: amino acid's net charge, hydropathy, average mass, Pk1 and Pk2, or van Der Waals volume, leads to an average 70% chance that the amino acid replacement does not significantly affect the protein. Furthermore, we observed that properties themselves, regardless of mutations, already bear a significant correlation with pathogenicity. For instance, the average mass of the original amino acids surrounding the position were the variant will take place (7 amino acid window) is a good indicator of pathogenicity (see Figure 5.7e), with larger amino acids having increasingly higher chances of being pathogenic.

We also found no correlation between pathogenicity and the predicted secondary structure affected by the variant. That is, α-helices and β-strands have only slightly more chance of pathogenicity than random coils (55% *vs.* 46%, n = 8962 *vs.* n =

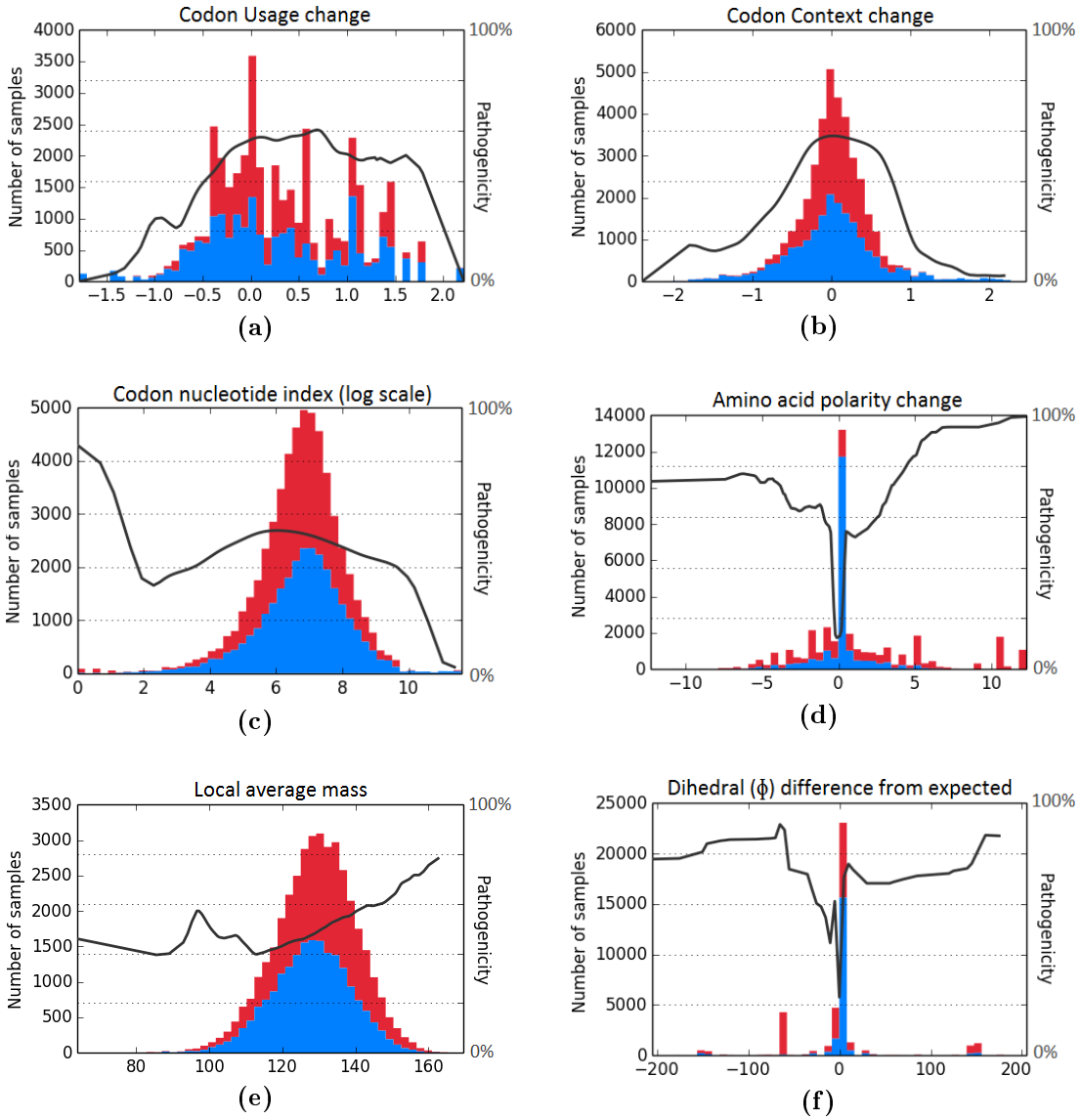


Figure 5.7 Features and pathogenicity distributions. Stacked histograms are shown for several features, showing neutral variants in blue and pathogenic variants in red (the horizontal axis represent the values of each feature). The black line indicates the expected probability of pathogenicity at each value of the feature. Some features show very pronounced effects, such as changes in the amino acid polarity (**d**), where variants quickly become largely pathogenic.

769). However, we detected an unexpected tendency for variants to not occur inside β-strands regions (only 1.4% of variants), which is not observed with α-helices (16.1% of variants), indicating a selective pressure to avoid disrupting β-strands. Moreover, finding a variant in a transition region (24.6% of variants) is more likely than inside a β-strand region.

When analyzing the changes in expected dihedral angles between the affected amino acids and their neighbors we observed agglomeration regions where a large number of variants clustered. For instance, 10% of the variants have Phi dihedral angles changing between -55° and -65° (Figure 5.7f), with 98% of them being pathogenic. Likewise, 10% of the variants in the Psi angle with the previous amino acid (and 13% with the following amino acid) have -20° to -30° Psi angles rotation changes, with 88% of them being pathogenic. The Ramachandran features revealed to have a high discriminatory value, and it becomes specially visible when plotting Phi against Psi angle differences, discerning pathogenic from neutral variants, as depicted in Figure 5.8.

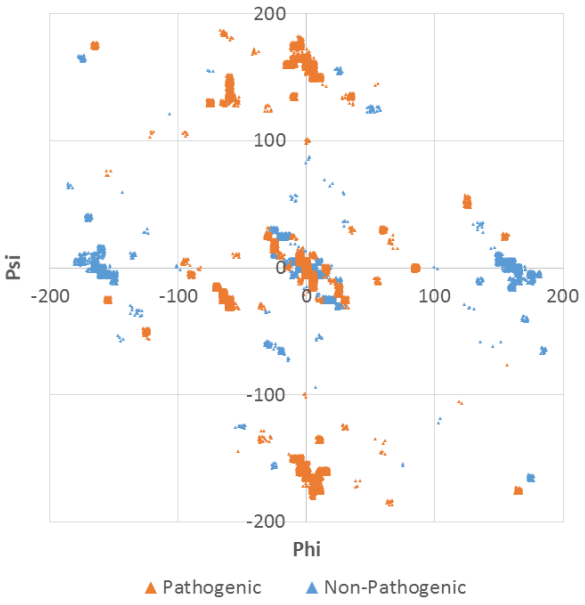


Figure 5.8 Phi vs. Psi angle difference plot. The dihedral angles differences between original and mutated amino acids are shown, coloring pathogenic and neutral variants. Islands of isolated variants with a single classification are common and clearly visible. It should be noted that though some scattering was applied to ease the understanding of the amount of variants in each region, the central zone around zero degrees has a far greater number of variants.

Moreover, by using the Gene Ontology to categorize all the affected proteins in our dataset, we were able to assess how the protein function and location relate to pathological issues. After training a linear SVM classifier we used its feature weights to measure the importance of each Gene Ontology term and found that the presence (or otherwise) of annotations under catalytic activity, developmental process, transporter activity, cell part, and cellular process are the most associated with pathogenicity (Table 5.4). For example, a mutated protein that is not annotated with cellular process or cell part has only 19.5% chance of being pathogenic; while mutations affecting proteins annotated with developmental process have a 68% chance of pathogenicity.

	Pathogenicity probability when	
Gene Ontology Base Term	**Not annotated**	**Annotated**
catalytic activity	43.6%	61.2%
developmental process	38.0%	68.0%
transporter activity	48.3%	63.8%
cell part	19.5%	53.9%
cellular process	19.5%	57.5%
extracellular region part	48.0%	63.1%
metabolic process	38.2%	61.4%
structural molecule activity	48.2%	65.9%
multicellular organismal process	31.2%	67.4%
membrane-enclosed lumen	47.5%	72.6%

Table 5.4 Most significant ontology terms according to the linear SVM classifier. The 10 most significant gene ontology features are shown (ordered by feature importance, the most important on top). Each feature represents all GO terms that derive (children) from that term. The ratio of pathogenic variants for each feature reveals the probability of a term being related to pathogenicity. There is also a pathogenicity probability associated with the lack of annotation, for example when cell parts are not directly affected (i.e. the affected protein is not annotated with cell part or any child term) there is a low chance (19.5%) that the variant is pathogenic. It should be noted that the importance of a term here is measured by its weight inside the SVM classifier, and not directly by the probability of pathogenicity.

5.3.2 PATHOGENICITY PREDICTION

Prior to testing classifiers, we pre-processed our input data set to reduce dissimilarities between features with large scale differences and reduce noise. For that, we tested several pre-processing approaches and combinations of these, as seen in Figure 5.9. For each pre-processing test, we used an SVM with the RBF kernel and performed a 5-fold cross-validation. The pre-processing was made only to the train data, and

a pre-processing model was recorded and applied to the test data, in order to avoid any biases and information overflowing between the training and testing phases. The f-measure evaluation of the model indicated the value of the pre-processing strategy used. Scaling the features and then normalizing the magnitude of each sample was found to be the best combination.

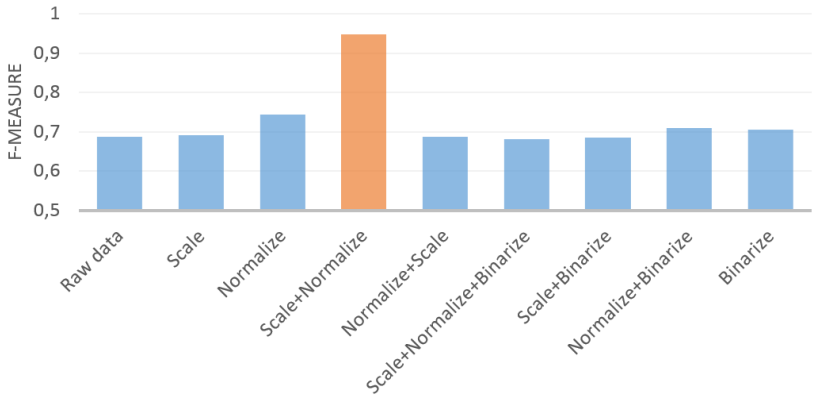


Figure 5.9 Comparison of pre-processing techniques.

To find the best machine learning classification method and overcome the issue with the no-free-lunch theorem, we sought to test several of the most popular approaches able to perform binary classification. Our goal was to build a model able to distinguish pathogenic from neutral human variants, and our hypothesis was that mRNA features are important and contribute significantly to the classification process. Among all the tested methods we found that using Support Vector Machines with a Radial Basis Function kernel yielded the highest performance, as measured by accuracy, F-measure and MCC (Figure 5.10). The tests were performed using a 5-fold cross-validation, and therefore a fifth of data is not used to train the model, which lowers the score marginally. Using a 100-fold cross-validation increased the RBF results to an average of 0.96 F-measure.

In order to reduce the dimensionality of our dataset and decrease its amount of noise, we tested several matrix decomposition methods, whereby the dataset is decomposed into linearly independent components, and the most important components are then selected for training-testing. Effectively, this is a form of feature selection, though the final features don't directly correspond to the initial ones. A comparison between the tested decomposition methods is shown in Figure 5.11, where the initial dataset was decomposed into 20 to 140 linearly independent components using 5

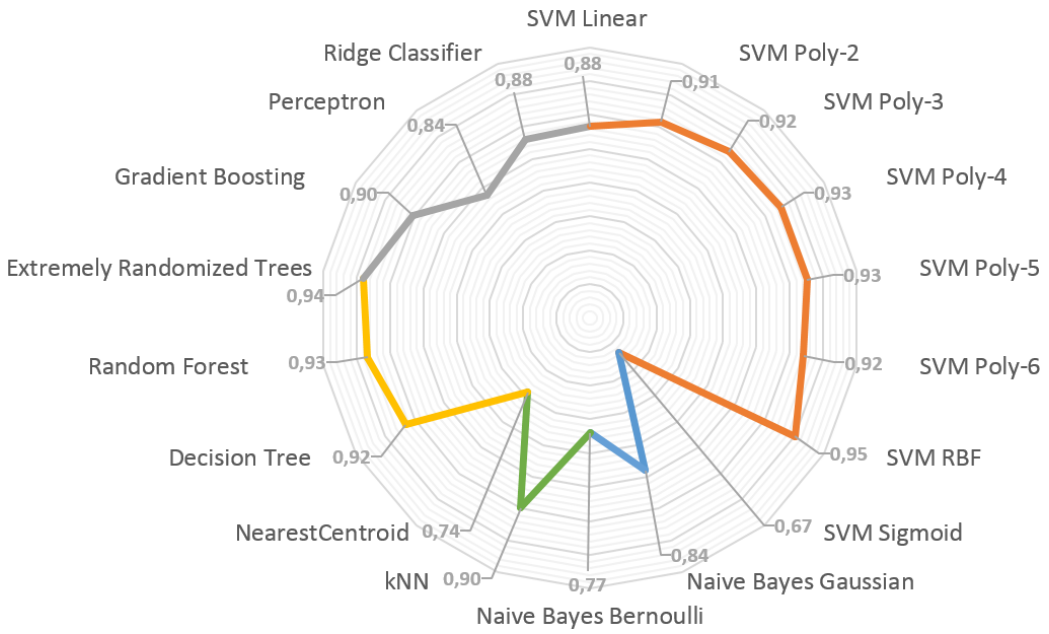


Figure 5.10 Comparison of classifiers performance. The F1-measure was used to compare the 18 approaches and their variations. Support Vector Machines and Tree-based classifiers behave generally better, with the RBF kernel having the largest performance.

methods.

For the decomposition tests we used a smaller, yet also balanced, dataset with 20 thousand variants, and only a 2-fold cross-validation, so the results could be computed in considerably less time. We assumed, thereby, that the results obtained with the smaller dataset and inferior validation could be extrapolated to the full dataset and complete validation. From our results we observed that using Latent Semantic Analysis (LSA, also known as truncated Singular Value Decomposition) and Principal Component Analysis (PCA) we can indeed reduce the dimension of the input data and still obtain results as good or even better (number of components = 100) than the baseline results (i.e. without matrix decomposition). By using the LSA strategy against the full dataset we confirmed a marginal improvement in accuracy, and a considerable improvement in training speed.

Yet another strategy to reduce the dimensionality of our dataset was to create a distance map based on the assessment of correlations between features, and remove those that were highly associated with each other. The reasoning behind this idea is that features with large correlations bring redundancy to the dataset, and are

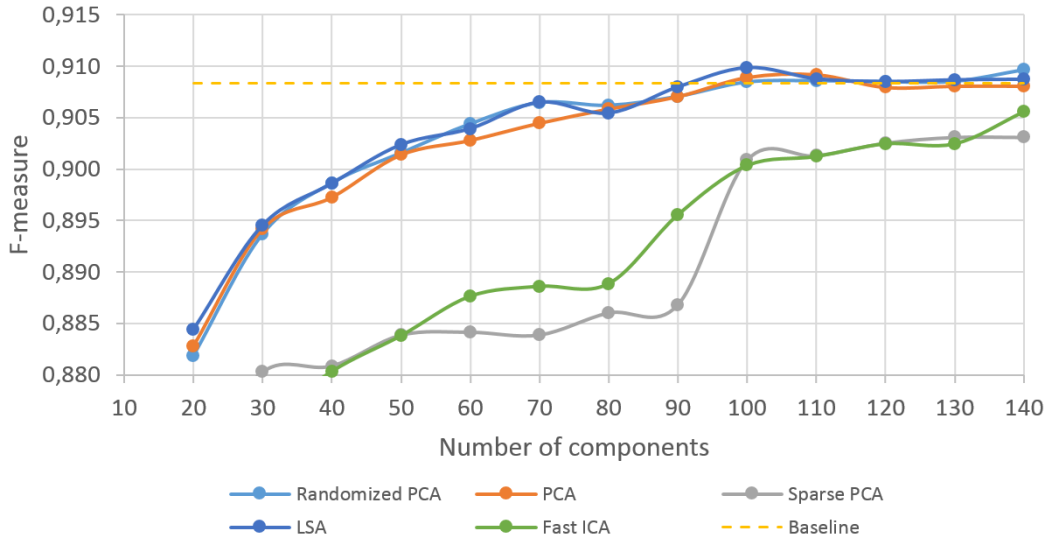


Figure 5.11 Comparison of matrix decomposition methods. For each approach, we limited the amount of resulting components to be used in the learning phase after the decomposition. We varied the number between 20 and 140, in steps of 10 components, and assessed its value from the f-measure results of the cross-validation. The yellow baseline represents the f-measure result without dimensionality reduction, and in some values it becomes surpassed by PCA (normal and randomized) and LSA.

possibly a source of noise, besides the fact that they do not add significant amounts of information when considering the downsides of having larger dimensions and longer learning times. In Figure 5.12, a color matrix shows the correlation between each pair of features, and also with the class. Several features are clearly dependent on each other.

We removed all redundant features whose Person's correlation is above 90% (both positive and negative wise), such as between the original amino acid's hydropathy and its polarity, or between the original gene's average RSCU and the ratio of hidden stop codons. Additionally, we removed Gene Ontology features that were too sparse and brought a negligible amount of information to the dataset: for example the term GO:0005623 was only positive in tree instances of the dataset and null in the remaining. After removing the redundancy, the dataset became reduced to 122 features, with features less linearly dependent on each other. However the resulting performance did not improve (decreased in 0.04%). This was likely due to the loss in information being superior to the gain in having a more consistent dataset.

In order to assess the predictive power of mRNA related characteristics, we re-

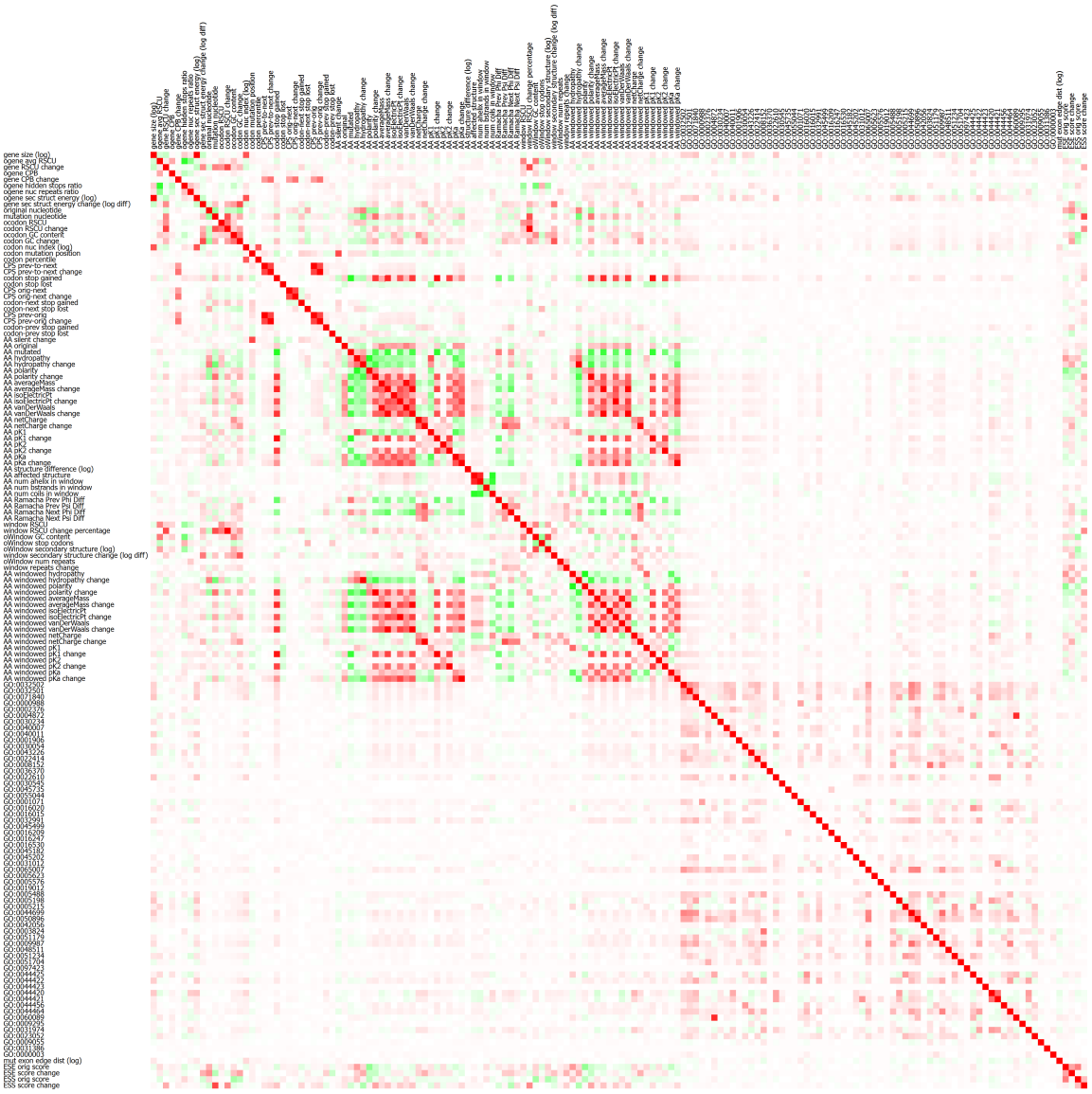


Figure 5.12 Feature correlations distance matrix. Rows and columns represent all the features. The color of each square indicates the amount of correlation between the two intersecting features. Red indicates a positive correlation, while green a negative one.

moved protein related features from the dataset (i.e. features regarding protein structure, function, or amino acid physicochemical property changes), thereby excluding features frequently described in literature, and performed a 20-fold cross-validation experiment. The model still reached over 92% accuracy, which is already higher than current pathogenicity prediction efforts. By further removing Gene Ontology features, hence keeping only mRNA features, the model achieved 89% accuracy, reflecting the strength and importance of mRNA characteristics in classifying the pathogenicity of variants. Figure 5.13 depicts the performance of the dataset when isolating each of the main types of features (gene features, role features, and protein features).

Figure 5.13 Performance by type of feature. The value of each of the three major types of features is assessed by training and testing with only one type. AA features are features related with amino acids and the protein structures.

We evaluated the prediction performance of our final model using 6 different assessment techniques: accuracy, f1-measure, sensitivity, precision, MCC and the area under the ROC curve. The results are shown in Table 5.5. Since the datasets we used in the learning schemes were always balanced, accuracy does not bear the problem of being biased towards one of the classes.

Accuracy	F1-Measure	Sensitivity	Precision	MCC	AUROC
0,95	0,95	0,97	0,94	0,91	0,99

Table 5.5 Performance indicators. All indicators vary between 0 and 1, except for MCC which varies between -1 and 1.

We further confirmed our model's ability to capture the dataset's structure and avoid over-fitting by using the permutation strategy described in the methods section,

with 100 permutations, to estimate the null hypothesis. As expected, we found that the null hypothesis was observing a 0.499 f-measure (0.5 is the expected from a random classification) with a standard deviation of 0.007 (see Figure 5.14), and therefore our model's p-value was lower than 0.01, which makes it statistically relevant and enough to reject the null hypothesis [136].

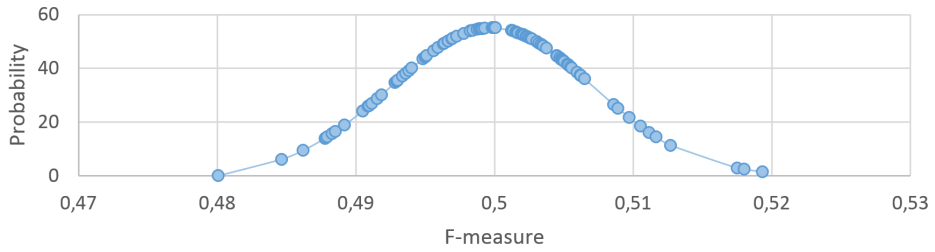


Figure 5.14 Distribution of results in a permutation test. Each point represents a test where the classes were randomly permuted (n = 100 tests). As expected, the randomization leads to a result of 0.5 f-measure being the most expected observation. Our model yields a 0.96 f-measure, far away from these values, which leads to a probability of finding that result (or better) by chance of less than 0.01 (p-value).

To extrapolate the value of each feature we employed information gain, recursive feature elimination and the feature weights of linear SVMs. After scaling all measures to the range 0-1 we ordered features by their importance according to the sum of the measures (the 30 most important are depicted in Figure 5.15).

It becomes clear that the binary feature indicating if a mutation causes or not an amino acid replacement is a factor of utmost importance, as it was awarded the maximum score. This can be explained by the fact that the majority of the silent mutations in the dataset (which represents about 28% of the instances) is neutral, and is related with the lack of publishing of silent pathogenic variants in literature. It is also visible that the majority of the top features are related with the protein (AA, amino acid), with only the presence of a few GO terms and the mRNA secondary structure energy and gaining a stop codon. The contradiction with the results shown above (Figure 5.13) where features of the mRNA have a larger impact can be explained by the information bore by individual mRNA features *versus* the information contained by the group of protein-related features. Gene features bear less information individually (and are less correlated with the class), but the classifier transforms and combines them in an optimal way, since the classification was made resorting an RBF kernel and not a linear one (which was used to assess the importance of the features). On the other hand, protein features (specially amino acid features) are more directly

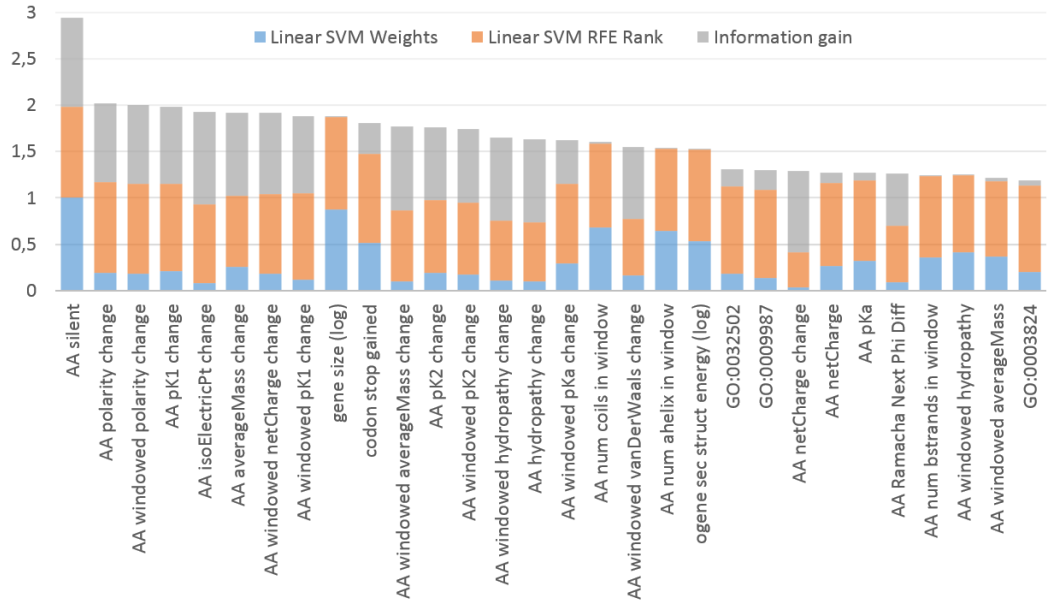


Figure 5.15 Feature importances using several assessments.

connected with pathogenicity but are significantly more redundant among themselves (as seen in the correlations matrix in Figure 5.12).

Finally, comparing the overall accuracy of our method with that of the most popular methods in literature (Figure 5.16) we find a significant advancement of 10% over the top methods for general pathogenicity prediction.

5.3.3 SIBYL

In order to ease and promote the use of our models we created Sibyl, a web-based tool available at bioinformatics.ua.pt/sibyl, that resorts to a previously trained model to perform online predictions (see Figure 5.17). Though the number of features required to make a pathogenicity assessment is large, the user is only required to supply two items: the reference ID for the mRNA where a mutation occurred, and the annotation of the mutation, which indicates where the mutation was found in the sequence and what was the substitution. The algorithms runs in the background to extract all features using the given data, build an input vector, and use the created SVM model to assess if the entry is pathogenic or neutral. Alternatively, the user can inform the system about the reference ID of an existing SNP in the online SNPdb, which will be then fetched and analyzed likewise.

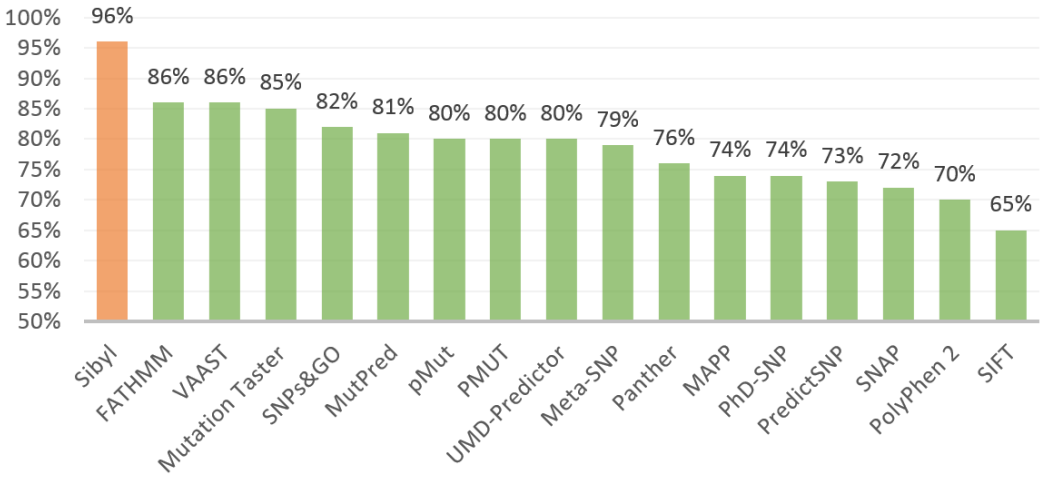


Figure 5.16 Comparison of popular methods accuracy. We included only general prediction methods, and not trait specific such as AUTO-MUTE.

Figure 5.17 Screenshot of the Sibyl online system. The first version encompasses only a basic input of transcript ID and variant annotation, or a reference SNPdb ID.

An application programming interface was also created to access Sibyl's functionality and enable the creation of more applications related with pathogenicity analysis through the use of our methods. A RESTful API is available to the outside, using the following resource code:

*http://bioinformatics.ua.pt/sibyl/api/predict/**{variant_annotation}**/**{transcript_id}***

The variant annotation needs to be according to the Human Genome Variation Society (HGVS) format, for example *c.138C>T*. The mRNA transcript reference ID is the identification used within the National Center for Biotechnology Information (NCBI), for example *NM_003761.4*. These are widely used standards which facilitate the access to available resources.

5.3.4 VARIOBOX

We have seen an increased interest and focus on genetic sequence evaluation and variant analysis in the past few years, as the number of existing locus specific databases (LSDBs, built from tools such as LOVD [61] and UMD [22]) rises, and solutions for the integration of individual genomic data sprout to aid in the organization of the large amounts of genetic data being created [109, 110].

Together with the increase in available genetic information is the increase in its use for diagnostic purposes and disease management. In fact, progress in this field is paving the way for a genetic-based medicine which demands new technology to handle individual genomic and variant profiles [71]. Also, the overwhelming quantity of patient genetic data emerging from sequencing laboratories suggests the need for integrated solutions that enable a more efficient use of variant knowledge for genetic research and patient-care scenarios.

Software such as MEGA4 [178], ANNOVAR [190] or SeqScape[5] streamline the initial sequence analysis process with the integration of alignment, variation inference and hypothesis testing capabilities. However, these tools lack essential functionalities regarding the annotation of input sequences and the interoperability with external systems. The latter is rather important, as we must enable adequate access to the abundant genetic data available in external resources.

Therefore, we developed Variobox, a new tool to support the patient sequence analysis workflow, integrating gene and variome analysis, including pathogenicity

[5](Life Technologies, http://www.lifetechnologies.com/)

prediction. Variobox comprises three main components to tackle the aforementioned genetics software challenges: **1)** Gene annotation: the selected gene reference sequence is annotated with variants from external databases and additional protein metadata from UniProt [11] and PDB [21]. Variants are analyzed using our models to infer their pathogenicity; **2)** Gene analysis: individual input sequences are shown in an advanced exploration interface, enabling navigation through the intronic/exonic regions of genes and the corresponding protein sequences; **3)** Gene comparison: the input sequence is compared to the reference sequence from LRG or RefSeq databases, enabling the automated detection of sequence variants and generation of corresponding standardized HGVS-compliant descriptions. A screenshot of the interface is depicted in Figure 5.18.

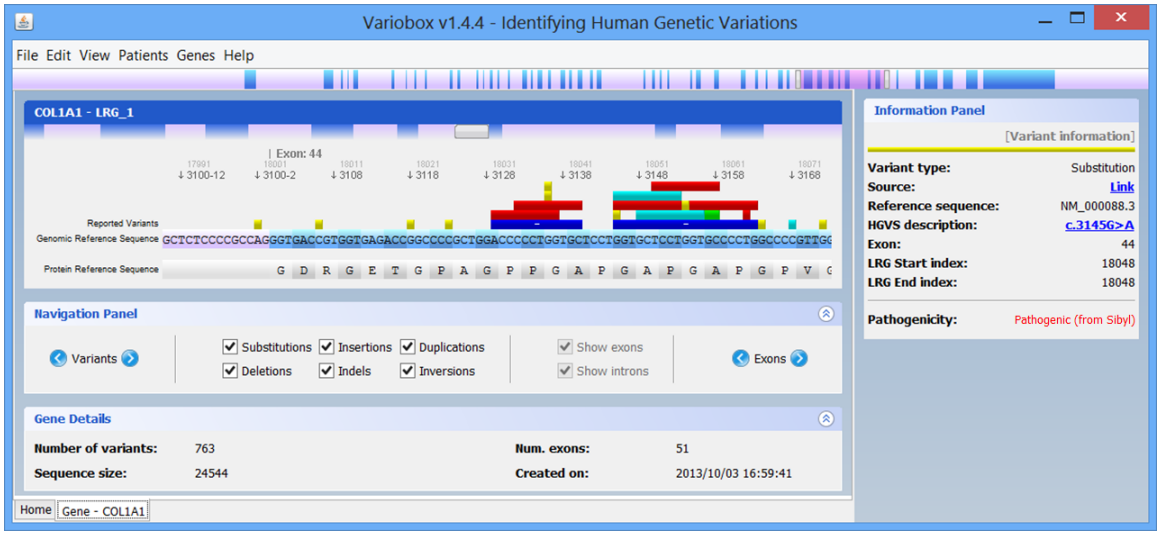


Figure 5.18 Screenshot of the Variobox interface. The main window of the application is depicted, with the working space on the left and the information panel on the right. The gene exploration interface is shown, with the three regions of navigation: a gene overview, showing exons and introns, and a resizable selector window; a sub-region exon-intron overview with a slider to further specify a region; a zoom-view of the nucleotide sequence showing codons, amino acids, corresponding variants and numbering, and the name of the exon. Colour distinguishes variants types: yellow are single nucleotide variants, red are deletions, green are insertions, cyan are duplications, pink are inversions, and grey are indels. The navigation panel is shown below, with controls to explore the gene and variants.

Variobox is built to provide intuitive visualization of sequences, presenting them in several levels of abstraction that range from an overall perspective of the gene (introns and exons) that allows selecting a window for further exploration, to a de-

tailed "zoom" perspective where nucleotides, amino acids and known variants are highlighted. Further details about each variant can be found in the information panel after selection, such as its position, type, data source, reference sequence, annotation, and an assessment of pathogenicity performed using the service provided by Sibyl.

We designed the software to fulfill the requirements of both genetic researchers and medical genetic laboratories to ease the step towards personalized genomics. Thus, it also comprises gene comparison and management features, such as the ability to load sequencing data in plain-text FASTA format or as SCF or AB1 sequencing chromatograms, and supports paired forward-reverse chromatogram reads, with automatic assembly of the final sequence.

Overall, Variobox works as a first approach to a unified and integrated environment that explores resources from the several levels of a patient genetic diagnosis scenario. It is available at bioinformatics.ua.pt/software/variobox.

5.4 CONCLUSIONS

Genetic variants not only dictate phenotypic differences between individuals, but are also the underlying cause of many disorders. Detecting, understanding, categorizing and associating human variants with disease phenotypes is becoming the standard process to accomplish personalized medicine. Starting with the Human Variome Project [154], which aims to collect, integrate and categorize variants at a global scale, several initiatives have made significant steps towards the standardization of genetic sequences and the availability of human variant annotations.

This recent focus on genetic sequence evaluation increases the demand for new variant analysis methods that assist clinical and laboratory geneticists, researchers and physicians. Moreover, substantial developments for medicine are now beginning to take place in the fields of genomics and proteomics, greatly assisted by the technology revolution of the past three decades, and mostly by the advancement in global communications. Transmitting knowledge and new discoveries on human gene sequences and variants on a global scale is now an effortless task. However, the problem is shifting towards the search and extraction for useful information and knowledge among the overwhelming volume of data that is being generated. Hence, the goal of personalized medicine is substantially hampered by our ability to explore and transform data into useful and informative shapes.

One of the largest searches in the last few years is the search for methods to extract knowledge from DNA data, and one of its greatest branches is finding hints within the genetic code to whether diseases exist or might manifest. Currently available patient variome data enables performing data mining and machine learning to that end, and several approaches have already tackled the task of predicting the pathogenicity of variants. Having that under consideration, we have set to contribute to this field in three-fold:

- Develop algorithms to analyze and extract information from the mRNA of target genes, and show the significance and importance of these genetic characteristics in the prediction of pathogenicity, as they are generally ignored in literature despite overwhelming evidence otherwise.
- Gather a dataset of characteristics based on literature hints and previous knowledge, and perform an analysis on the extracted characteristics to infer patterns of discernment between pathogenic and neutral variants that are relevant to the field.
- Develop a new machine learning methodology based on the extracted characteristics to perform pathogenicity prediction with high accuracy.

We found several substantial patterns in our collected data, and explored those patterns using statistical models to perform classification and create a new approach to pathogenicity prediction. When compared with other methods, our strategy achieves superior performances, and we've shown that a large part of the performance is related with evaluations of the gene and mRNA.

Our method was created by studying several machine learning approaches, preprocessing methods, dimensionality reduction methods, and by seeking to correctly evaluate our models and their statistical significance to avoid over-fitting and measure its ability to generalize. In the process, we also deeply analyzed our dataset and found significant patterns with biological meaning, such as pre-determined dihedral angles where pathogenicity is more likely to occur, or selective tendencies for variants to avoid affecting protein regions with β-strands. To promote the use of our method we created a simple web interface to allow accessing the classification model and perform predictions.

We further approached the patient genetic diagnosis pipeline by creating a first unified workflow that connects patient gene sequencing with available variant data,

genetic exploration abilities and pathogenicity evaluation.

6 | CONCLUSIONS

"Let us understand what our own selfish genes are up to, because we may then at least have the chance to upset their designs, something that no other species has ever aspired to do."
— Richard Dawkins, The Selfish Gene

Biopoiesis, the process whereby life is initially created, tries to explain how living beings could have been formed from common compounds on earth. Current theories and experimental data[1] suggest that very simple molecules naturally arose, called replicators, that had the ability to copy themselves into new similar molecules also with that ability, and so on. The fidelity with which those molecules replicated was not perfect, and thereby new molecules often contained differences, mutations, that made them different from their parents but, more importantly, conferred them characteristics that could impact their replication abilities either positively or negatively. Over time, replicators with a better ability to copy themselves and access surrounding resources became more abundant than those with insufficient fidelity to copy or incapacity to "fight" for resources. These molecules were not life as we know it, but bore important properties still common to living forms, such as replication and variation capabilities, which granted them evolvability. In fact, a laboratory created for the first time in 2009 a molecular system capable to self-replicate, evolve, and adapt to the environment, without being life-based [90].

According to the gene-centered view of evolution, these replicators were the primordials of what are now known as genes. As a consequence, since genes are arguably the main entity in humans, we are bound to evolve through the same mechanisms: replication (by reproduction) and variation (by cross-over and mutations). Compu-

[1]Such as the Miller-Urey experiment, where amino acids were naturally formed using the same conditions Earth had a few billion years ago.

tational heuristics such as genetic algorithms and simulated annealing, which were explored in chapter 3, make use of the same principles to find solutions that fit to a problem's constraints. Replication errors (variation) are therefore one of the main drivers behind evolution, and in humans that means they are the main driver behind not only our phenotypes but also our ability to dodge and fight harmful beings (e.g. disease causing bacteria) and other machinery whose goal is also replicating (e.g. viruses). One should not forget that we are invulnerable to the majority of organisms in our environment due to random replication errors that occurred in our ancestors. However, that evolutionary ability comes at the cost that the same replication errors are also responsible for causing disease. That can occur in two ways: either by creating a genetic defect that causes a disruption in the normal internal functionality of the organism, or by lack of functionality to cope with external agents (new bacteria strains, viruses, etc.). In fact, the Nobel prize laureate Paul Berg once stated that *"All human disease is genetic in origin"*.

It was only in the past few years that human variation became accessible, with the pace of genome sequencing accelerating (by 2010, with the cost of sequencing plummeting) and new computational technologies enabling its categorization and availability. Only now it has become feasible to start looking for patterns and create a global understanding of our genetic code and the meaning and impact of differences. Many new disease-causing mutations have already been identified [95, 131, 132, 184], and the impact those findings bring to the clinical area is gaining wide interest. The large amounts of genetic data currently being generated worldwide suggest that the bottleneck to knowledge discovery now belongs to the fields of computer science and in particular to bioinformatics.

6.1 HYPOTHESIS

As genes evolved, they also fine-tunned their structure and information to improve their fidelity and compete for survival. It is now known that several pieces of information within the genetic code are largely responsible for its performance, and the most clear example is the existence of a bias in the use of codons (codon usage). With the availability of genetic codes in digital format, it is possible to perform measurements and detect patterns related with the information carried by genes. Since this information is crucial to the organism and its functioning, it is only logical that replication errors altering it can have a large impact. Therefore, measuring alterations in the

information in genes becomes a viable path to detect pathogenic issues, and one that can be achieved algorithmically.

In this work we have sought to create computational methods to find, measure and manipulate molecular and genetic characteristics (genetic "secondary" information) and use the created approaches to predict the pathogenic outcomes of human mutations. For that, I used chapter 2 to contextualize the panorama of genetics by showing the importance of the gene and its translation in an organism, and review the main gene characteristics pointed in literature has having a large impact. For instance, I discussed the impact of codon usage, ramp effect, codon correlation effect, codon context, content of GC, hidden stop codons, repeated nucleotides, external interaction sites and the large importance of secondary structure formation in the mRNA. I have also shown several strategies used to analytically evaluate these characteristics. In the same chapter, I also described other characteristics, besides those affecting the mRNA, that might have an impact in pathogenicity. Several existing methods to predict pathogenicity mainly make use of protein homology, but also protein structural characteristics.

Chapter 3 describes how we pursued the idea of evaluating genes, by creating algorithms capable of performing assessments based on the gene characteristics discussed in chapter 2 (expression indicators). This chapter was then centered on the discussion of how to aggregate gene evaluations and employ them to manipulate genes, and on the challenges behind finding gene conformations for conflicting objectives. I also described how we employed a *Pareto archive* to find optimal genes for a goal. Manipulating genes based on expression indicators is one example of the applicability of gene evaluations to an end.

We then further explored one of the most intricate gene evaluations in chapter 4: the formation of mRNA secondary structures. Since current methods to assess the free energy of an mRNA molecule are time and memory expensive, they could not be used to perform massive evaluations, as needed to perform optimizations or evaluate the energetic changes occurring in many mutations. Therefore, we created a simplistic approach for the assessment of the minimum free energy of an mRNA molecule with a fast algorithm, while fitting the results of the algorithm to those of accurate measurement tools, to reduce the error.

Chapter 5 resulted as the culmination of the developed methods for evaluating genes, turning to the goal of predicting pathogenicity in human genes. To that end, we collected a considerable amount of curated variant data from online sources to

use as our gold standard, and sought to explore statistical models able to predict the pathogenicity of variants, by using the gene evaluations that were developed, together with other hints. We built a coherent dataset consisting of evaluations of the genes affected by the mutations we collected, with the reasoning that these evaluations serve as hints to what information carried by the gene was changed, and that a machine learning model could explore them to infer pathogenicity. We also explored other characteristics also related with the affected proteins, and hypothesized that Gene Ontology could provide functional annotations that indicate the role of the protein. By analyzing the final dataset we were able to find several relevant patterns that reveal biological tendencies that we consider important to understand the impact of variants. We then studied several models, data processing techniques, and optimization strategies to build a high performing classification model to predict pathogenicity.

Our hypothesis was that we could use mRNA-based observations and evaluations to tackle the pathogenicity prediction problem. In the process, we also sought to use the same evaluations to tackle different problems, such as gene optimization for heterologous expression and mRNA secondary structure manipulation.

6.2 OUTCOMES

In the process of creating strategies to evaluate and manipulate genes we achieved several meaningful results, which are now summarized as follows:

- From our review of literature we gathered a compilation of gene expression indicators that play an important role in organisms. These indicators were used for creating algorithms capable of evaluating genes and return a numeric assessment of the expression indicators.
- We built a framework of gene evaluation and optimization that allows combining the developed assessment algorithms and any new algorithms. We used multi-optimization strategies to build a methodology to explore the solution space of possible gene permutations, by exploiting the degeneracy of the genetic code.
- We tackled an important problem in the field of RNA by creating a new algorithm capable of rapidly finding the minimum free energy created by the formation of secondary structures in mRNA molecules, at the expense of a reduced accuracy but highly correlated. The created algorithm allowed building a

tool capable of optimizing mRNA molecules by maximizing or minimizing their minimum free energy. Since the algorithm is fast, it also allows performing a large number of evaluations.

- When analyzing the variant data that was extracted and measured using our algorithms, we found several patterns with a facilitated discrimination between pathogenic and neutral variants. For example: changes in codon usage below -0.5 have an 83% chance of being neutral; the proximity of the mutation to the edge of a gene exon is a very good indicator of pathogenicity with variants far from edges being increasingly neutral (up to 80%); or regions in the dihedral angle differences that accumulated pathogenic or neutral variant islands.

- We created a highly accurate method for predicting the pathogenicity of coding SNVs, based on the mRNA assessment algorithms, Gene Ontology, and protein evaluation algorithms. The method uses machine learning to learn to distinguish pathogenic from neutral single nucleotide variants in the coding regions of human genes, and its performance was tested using cross-validation verifications, which yielded a 96% accuracy, with a p-value of 0.01.

- From our machine learning prediction model we were able to determine which characteristics bear the most importance for the classifier to decide if a variant is likely to cause a disease. We observed that protein chemical change characteristics are the features which individually have the most weight, but mRNA features as a whole have a larger impact. This role of mRNA-related characteristics contradicts current trends in literature that mainly focus on homology and the conservation of the affected regions among species. Assuming that the importance of characteristics in the classifier translates into biological importance, these findings may be relevant for understanding and gaining novel insight into the role and impact of variants.

- Besides the computational approaches, and to streamline the work in research labs, three software applications were built or continued.

  - First, using the developed optimization and evaluation framework and the assessment algorithms, we evolved a practical solution (Eugene) for biologists to analyze and manipulate their genes through a usable interface. The tool is directed at redesigning genes for heterologous hosts using multiple simultaneous goals.

– Second, we have built a systems for the analysis of variants, Sibyl, which is a simple online interface to perform an assessment of pathogenicity for a given variant, making use of our trained models. To encourage using our strategy, along with Sibyl we also developed a RESTful interface to perform online predictions from within other tools and services.

– And third, Variobox, is a platform for the exploration, annotation and analysis of the human variome. This platform creates a first approach to the unification of patient genetic data, using information from gene sequencing along with knowledge present in online databases. Patient genes are thereby compared with their reference genes, and found and known mutations are shown along the sequences, allowing a quick assessment of genetic alterations in patients, their meaning as reported in literature, and their predicted pathogenic outcome from our prediction models.

6.3 FUTURE PERSPECTIVES

Along the methods, procedures, and results that were obtained in this work, there are numerous possible improvements and lines of work that were left behind in favor of achieving the final goal. These improvements might represent significant advances in several areas, and should not be discarded in future work. They are described as follows:

- For the multi-optimization of genes we used two strategies to explore the space of possible codon permutations, one of them to quickly return a good solution (simulated annealing), and another one to more deeply explore the space and attain a range of equally viable good solutions (genetic algorithm, using a *Pareto archive*) at the cost of a slower return. However, a better strategy would be to quickly explore the space until the *Pareto front* is found, and only then expand the search to find as many *Pareto optimal* solutions as possible. One form to tackle the problem that is to take advantage of the fact that we can often easily achieve points that are certain to belong to the *Pareto front*, the same way we can calculate how the ideal solution would look like (by finding the individual optimals). There are already several solutions to explore *Pareto fronts* by varying the weights of each individual goal, which could be adapted [185, 204].

- In creating our prediction model, we concentrated on the most common type of mutation: single nucleotide variants. However, as demonstrated in the second chapter, there are several other types of mutations. Performing the same analysis and creating prediction models for these mutations extends the range to which our approach can be used. Indeed, we have already created methods in Variobox to identify and annotate several types of mutations, and the majority of the features that were used can be adapted to other types. Furthermore, another possible and important development can be obtained by performing an analysis and creating a prediction algorithm for variants occurring outside coding regions (98% of the genetic code), which are now known to be mainly constituted of regulatory elements and transposable elements[2]. Also, while most of the work we developed is single-gene oriented, some of the most difficult obstacles in the state of the art lie in the inter-relations between mutations, even in different genes or non-coding regions. Contemplating the vast dependencies that occur among the several regions of the genetic code and their variation is an essential task to comprehend some of the most complex diseases, such as Alzheimer's.

- As mentioned earlier, the majority of current approaches for the prediction of pathogenicity relies on the use of homology to assess if the affected region is conserved within other species' genetic code. We have indeed used and tested a basic version of this approach and tryed to improve our model's performance, by including the values of the BLOSUM matrices [78] which foretell the probability of an amino acid being replaced by another in a conserved region. However we could not find a performance improvement, and the additional dimensions were not desirable. Nonetheless, we do not discard the value of using homology to reveal the importance of a region and infer the impact of a mutation in it. This assumption can be extended to the evaluation of the importance of the affected region, either by homology or by investigating the function of the region, for instance by looking for known protein domains. The use of homology can be specially useful when extending Sibyl to study specific diseases.

- The definition under which researchers are reporting variants as pathogenic is too broad: some consider pathogenic as affecting the stability of a protein and other consider pathogenic as disease-causing. We tried as much as possible to

[2]Replicators that created copies of themselves within the genome, and/or were placed there by retroviruses ages ago.

discern both, and include only those related with disease-causing, but there is still some lack of standardization. It is important that variants are reported in a concise format, and it is also important that they are reported independently of their origin. For instance, we could only retrieve 223 silent pathogenic variants, which hampers discerning pathogenic from neutral in silent variants. Also, as human genomic variation data becomes increasingly available, it is important to continue to update models, and specialize them to tackle more specific matters instead of general pathogenicity. For instance, if the available data allows it, models should be trained for populations with different ancestries (European, Asian, African, etc.) or different phenotypes (training for diseases).

- To correctly follow the scientific method, comparison with other tools should be performed on common grounds. For instance, we used the performance results reported in literature by the own authors of methods or by independent replication of the experiments when available. However, the lack of a systematic and standardized form to access the methods and to perform independent tests undermines the comparison. Also, the majority of the methods used their own collected variants, and though they supply the used dataset, since most methods are protein oriented, the variants are only available in amino acid mutation formats, and not in nucleotide mutation formats. Recently, VariBench [158], a variant database to benchmark pathogenicity prediction systems, was created specifically to tackle this issue. To correctly compare our methodology with other's, a common dataset such as those provided by VariBench should be used.

As a general guideline, our work is included in the whole landscape of genetic analysis: from sequencing the DNA of a patient to creating a useful analysis and diagnosis for medical practitioners and, ultimately, to prevention. The pipeline we mentioned in the first chapter is far from being holistic and complete. Instead, many other variables and entities must be considered, and specially data must still be directed to where it is needed and transformed into information and knowledge, which is one of the main goals of bioinformatics.

Data from ancestors, for instance a patient's parents, the origin of an allele and its frequency in its original population, and further comprehension of the functioning (and breakdown) of the whole genome are but examples of bits of information that must be included in developed tools and systems for a patient diagnosis pipeline.

6.4 FINAL REMARKS

After the publication of the first complete human genome, an excellent article was published in the Nature journal by Francis Collins *et al.* [43], stating that:

> *"The genomic era is now a reality. (...)*
> *Interwoven advances in genetics, comparative genomics, high throughput biochemistry and* ***bioinformatics*** *are providing biologists with a markedly improved repertoire of research tools that will allow the functioning of organisms in health and disease to be analyzed and comprehended at an unprecedented level of molecular detail."*

Indeed, the identification of the etiology of human diseases was once an herculean task requiring many years of research and financing. Nowadays the procedure is shifting, and identifying the causes for a disease requires only enough genomic data to perform data mining and find the differences between populations with and without the phenotype. With this setup, being diagnosed is as hard as comparing our genome with available data to find similarities.

Ultimately, progress in health, genetics and biotechnology will walk hand-in-hand, as developments in the biomedical domain require more technological advancements, which in turn will result in more developments. Eventually, these advancements have a vast impact in human life. The computational study of genes, their functionality, and the effects of the resulting proteins in organisms, is a goal that in its course has already led to human benefits.

I envision a future where it is a common procedure to computationally analyze the DNA of a newborn. Or even before birth. Having a complete model of our personal genome and its behavior, through a system whose bases we lay in this work, it is easy to picture an evolution of healthcare and personalized medicine in the sense of fully understanding what is happening in our organism at any time, prepare adequate treatment beforehand, and even manipulating other organisms to our own benefit.

BIBLIOGRAPHY

[1] G. R. Abecasis, D. Altshuler, A. Auton, L. D. Brooks, R. M. Durbin, R. a. Gibbs, M. E. Hurles, and G. a. McVean. A map of human genome variation from population-scale sequencing. *Nature*, 467(7319):1061–73, Oct. 2010.

[2] G. R. Abecasis, A. Auton, L. D. Brooks, M. a. DePristo, R. M. Durbin, R. E. Handsaker, H. M. Kang, G. T. Marth, and G. a. McVean. An integrated map of genetic variation from 1,092 human genomes. *Nature*, 491(7422):56–65, Nov. 2012.

[3] I. Adzhubei, D. M. Jordan, and S. R. Sunyaev. Predicting functional effect of human missense mutations using PolyPhen-2. *Current protocols in human genetics / editorial board, Jonathan L. Haines ... [et al.]*, Chapter 7:Unit7.20, Jan. 2013.

[4] H. Akhtar, S. Akhtar, S. U. Jan, A. Khan, N. U. S. S. Zaidi, and I. Qadri. Over expression of a synthetic gene encoding interferon lambda using relative synonymous codon usage bias in Escherichia coli. *Pakistan journal of pharmaceutical sciences*, 26(6):1181–8, Nov. 2013.

[5] E. Andrianantoandro, S. Basu, D. K. Karig, and R. Weiss. Synthetic biology: new engineering rules for an emerging discipline. *Molecular systems biology*, 2:2006.0028, 2006.

[6] M. Andronescu, A. P. Fejes, F. Hutter, H. H. Hoos, and A. Condon. A new algorithm for RNA secondary structure design. *Journal of Molecular Biology*, 336(3):607–624, 2004.

[7] C. B. Anfinsen, E. Haber, M. Sela, and F. H. White. The kinetics of formation of native ribonuclease during oxidation of the reduced polypeptide chain. *Pro-*

ceedings of the National Academy of Sciences of the United States of America, 47:1309–1314, 1961.

[8] E. Angov, C. J. Hillier, R. L. Kincaid, and J. a. Lyon. Heterologous protein expression is enhanced by harmonizing the codon usage frequencies of the target gene with those of the expression host. *PLoS One*, 3(5):e2189, Jan. 2008.

[9] E. Angov, P. M. Legler, and R. M. Mease. Adjustment of Codon Usage Frequencies by Codon Harmonization Improves Protein Expression and Folding. In J. Evans Thomas C. and M.-Q. Xu, editors, *Heterologous Gene Expression in E.coli SE - 1*, volume 705 of *Methods in Molecular Biology*, pages 1–13. Humana Press, 2011.

[10] S. E. Antonarakis and J. S. Beckmann. Mendelian disorders deserve more attention. *Nature reviews. Genetics*, 7:277–282, 2006.

[11] R. Apweiler, A. Bairoch, C. H. Wu, W. C. Barker, B. Boeckmann, S. Ferro, E. Gasteiger, H. Huang, R. Lopez, M. Magrane, M. J. Martin, D. A. Natale, C. OâĂŹDonovan, and N. Redaschi. UniProt: the Universal Protein knowledgebase. *Nucleic acids research*, 32(suppl 1):D115–D119, 2004.

[12] T. W. ATHAN and P. Y. PAPALAMBROS. A NOTE ON WEIGHTED CRITERIA METHODS FOR COMPROMISE SOLUTIONS IN MULTI-OBJECTIVE OPTIMIZATION, 1996.

[13] O. T. Avery, C. M. Macleod, and M. McCarty. Studies on the chemical nature of the substance inducing transformation of pneumococcal types: induction of transformation by a desoxyribonucleic acid fraction isolated from pneumococcus type III. *The Journal of experimental medicine*, 79:137–158, 1944.

[14] A. Avihoo, A. Churkin, and D. Barash. RNAexinv: An extended inverse RNA folding from shape and physical attributes to sequences. *BMC bioinformatics*, 12(1):319, 2011.

[15] S. Baik, K. Inoue, M. Ouyang, and M. Inouye. Significant bias against the ACA triplet in the tmRNA sequence of Escherichia coli K-12. *Journal of bacteriology*, 191:6157–6166, 2009.

[16] R. L. Baldwin. Energetics of protein folding. *Journal of molecular biology*, 371(2):283–301, Aug. 2007.

[17] R. Bellman. On the Theory of Dynamic Programming. *Proceedings of the National Academy of Sciences of the United States of America*, 38:716–719, 1952.

[18] J. Bendl, J. Stourac, O. Salanda, A. Pavelka, E. D. Wieben, J. Zendulka, J. Brezovsky, and J. Damborsky. PredictSNP: Robust and Accurate Consensus Classifier for Prediction of Disease-Related Mutations. *PLoS Computational Biology*, 10, 2014.

[19] D. A. Benson, K. Clark, I. Karsch-Mizrachi, D. J. Lipman, J. Ostell, and E. W. Sayers. GenBank. *Nucleic acids research*, 42:D32–7, 2014.

[20] K. Bentele, P. Saffert, R. Rauscher, Z. Ignatova, and N. Blüthgen. Efficient translation initiation dictates codon usage at gene start. *Molecular Systems Biology*, 9(1), June 2013.

[21] H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov, and P. E. Bourne. The protein data bank. *Nucleic acids research*, 28(1):235–242, 2000.

[22] C. Béroud, G. Collod-Béroud, C. Boileau, T. Soussi, and C. Junien. UMD (Universal mutation database): a generic software to build and analyze locus-specific databases. *Human mutation*, 15(1):86–94, 2000.

[23] C. C. Bigelow. On the average hydrophobicity of proteins and the relation between it and protein structure. *Journal of theoretical biology*, 16:187–211, 1967.

[24] E. Bingham and A. Hyvärinen. A fast fixed-point algorithm for independent component analysis of complex valued signals. *International journal of neural systems*, 10:1–8, 2000.

[25] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[26] E. Bossy-Wetzel, R. Schwarzenbacher, and S. A. Lipton. Molecular pathways to neurodegeneration. *Nature medicine*, 10 Suppl:S2–S9, 2004.

[27] D. Brégeon, V. Colot, M. Radman, and F. Taddei. Translational misreading: a tRNA modification counteracts a+ 2 ribosomal frameshift. *Genes & Development*, 15(17):2295, 2001.

[28] Y. Bromberg and B. Rost. SNAP: predict effect of non-synonymous polymorphisms on function. *Nucleic acids research*, 35:3823–3835, 2007.

[29] N. a. Burgess-Brown, S. Sharma, F. Sobott, C. Loenarz, U. Oppermann, and O. Gileadi. Codon optimization can improve expression of human genes in Escherichia coli: A multi-gene study. *Protein expression and purification*, 59(1):94–102, May 2008.

[30] A. Busch and R. Backofen. INFO-RNA - a server for fast inverse RNA folding satisfying sequence constraints. *Nucleic acids research*, 35(suppl 2):W310—-W313, 2007.

[31] G. Cannarrozzi, N. N. Schraudolph, M. Faty, P. von Rohr, M. T. Friberg, A. C. Roth, P. Gonnet, G. Gonnet, and Y. Barral. A role for codon order in translation dynamics. *Cell*, 141:355–367, 2010.

[32] E. Capriotti, R. B. Altman, and Y. Bromberg. Collective judgment predicts disease-associated single nucleotide variants. *BMC genomics*, 14 Suppl 3:S2, 2013.

[33] E. Capriotti, R. Calabrese, and R. Casadio. Predicting the insurgence of human genetic diseases associated to single point protein mutations with support vector machines and evolutionary information. *Bioinformatics (Oxford, England)*, 22:2729–2734, 2006.

[34] E. Capriotti, R. Calabrese, P. Fariselli, P. L. Martelli, R. B. Altman, and R. Casadio. WS-SNPs&GO: a web server for predicting the deleterious effect of human protein variants using functional annotation. *BMC genomics*, 14 Suppl 3:S6, 2013.

[35] E. Capriotti, N. L. Nehrt, M. G. Kann, and Y. Bromberg. Bioinformatics for personal genome interpretation. *Briefings in bioinformatics*, 13(4):495–512, July 2012.

[36] D. Chasman and R. M. Adams. Predicting the functional consequences of non-synonymous single nucleotide polymorphisms: structure-based assessment of amino acid variation. *Journal of molecular biology*, 307:683–706, 2001.

[37] D. Chen and D. E. Texada. Low-usage codons and rare codons of Escherichia coli. 10(c):1–12, 2006.

[38] H. Chial. Rare Genetic Disorders: Learning About Genetic Disease Through Gene Mapping, SNPs, and Microarray Data. *Nature Education*, 1:1, 2008.

[39] B. K.-S. Chung and D.-Y. Lee. Computational codon optimization of synthetic gene for protein expression. *BMC systems biology*, 6(1):134, Jan. 2012.

[40] A. Chursov, D. Frishman, and A. Shneider. Conservation of mRNA secondary structures may filter out mutations in Escherichia coli evolution. *Nucleic acids research*, 41:7854–60, 2013.

[41] J. R. Coleman, D. Papamichail, S. Skiena, B. Futcher, and S. Mueller. Virus Attenuation by Genome-Scale Changes in Codon Pair Bias. 320(5884):1784–1787, 2009.

[42] J. R. Coleman, D. Papamichail, S. Skiena, B. Futcher, E. Wimmer, and S. Mueller. Virus attenuation by genome-scale changes in codon pair bias - supporting material. *Science*, 320(5884):1784, 2008.

[43] F. S. Collins, E. D. Green, A. E. Guttmacher, M. S. Guyer, and U. S. N. H. G. R. Institute. A vision for the future of genomics research: a blueprint for the genomic era. *Nature*, 422:835–847, 2003.

[44] F. S. Collins, M. S. Guyer, and A. Chakravarti. Variations on a theme: cataloging human DNA sequence variation. *Science*, 278:1580–1, 1997.

[45] I. H. G. S. Consortium. Initial sequencing and analysis of the human genome. *Nature*, 409:860–921, 2001.

[46] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20:273–297, 1995.

[47] F. H. CRICK. On protein synthesis. *Symposia of the Society for Experimental Biology*, 12:138–163, 1958.

[48] J. F. Curran, E. S. Poole, W. P. Tate, and B. L. Gross. Selection of aminoacyl-tRNAs at sense codons: the size of the tRNA variable loop determines whether the immediate 3' nucleotide to the codon has a context effect. *Nucleic acids research*, 23:4104–4108, 1995.

[49] I. Das and J. E. Dennis. A closer look at drawbacks of minimizing weighted sums of objectives for Pareto set generation in multicriteria optimization problems, 1997.

[50] M. H. de Smit and J. van Duin. Control of Translation by mRNA Secondary Structure in Escherichia coli: A Quantitative Analysis of Literature Data. *Journal of molecular biology*, 244(2):144–150, 1994.

[51] K. Deb. Multi-objective optimization using evolutionary algorithms: an introduction. In *Multi-objective evolutionary optimisation for product design and manufacturing*, pages 1–24. 2011.

[52] K. A. Dill, S. B. Ozkan, M. S. Shell, and T. R. Weikl. The protein folding problem. *Annual review of biophysics*, 37:289–316, 2008.

[53] M. dos Reis, R. Savva, and L. Wernisch. Solving the riddle of codon usage preferences: a test for translational selection. *Nucleic acids research*, 32:5036–5044, 2004.

[54] R. C. Edgar. MUSCLE: a multiple sequence alignment method with reduced time and space complexity. *BMC bioinformatics*, 5(1):113, 2004.

[55] A. Eyre-Walker and M. Bulmer. Reduced synonymous substitution rate at the start of enterobacterial genes. *Nucleic acids research*, 21(19):4599–4603, 1993.

[56] J. L. Fauchère, M. Charton, L. B. Kier, A. Verloop, and V. Pliska. Amino acid side chain parameters for correlation studies in biology and pharmacology. *International journal of peptide and protein research*, 32:269–278, 1988.

[57] T. Fawcett. An introduction to ROC analysis. *Pattern Recognition Letters*, 27:861–874, 2006.

[58] G. H. Fernald, E. Capriotti, R. Daneshjou, K. J. Karczewski, and R. B. Altman. Bioinformatics challenges for personalized medicine. *Bioinformatics (Oxford, England)*, 27(13):1741–8, July 2011.

[59] A.-M. Fernandez-Escamilla, F. Rousseau, J. Schymkowitz, and L. Serrano. Prediction of sequence-dependent and mutational effects on the aggregation of peptides and proteins. *Nature biotechnology*, 22(10):1302–6, Oct. 2004.

[60] I. Fokkema, J. T. den Dunnen, and P. E. M. Taschner. LOVD: Easy creation of a locus-specific sequence variation database using an "LSDB-in-a-box" approach. *Human Mutation*, 26(2):63–68, 2005.

[61] I. F. A. C. Fokkema, P. E. M. Taschner, G. C. P. Schaafsma, J. Celli, J. F. J. Laros, and J. T. den Dunnen. LOVD v. 2.0: the next generation in gene variant databases. *Human mutation*, 32(5):557–563, 2011.

[62] M. Y. Frédéric, M. Lalande, C. Boileau, D. Hamroun, M. Claustres, C. Béroud, and G. Collod-Béroud. UMD-predictor, a new prediction tool for nucleotide substitution pathogenicity – application to four genes: FBN1, FBN2, TGFBR1, and TGFBR2. *Human mutation*, 30:952–959, 2009.

[63] A. Fuglsang. Codon optimizer: a freeware tool for codon optimization. *Protein expression and purification*, 31(2):247–249, 2003.

[64] W. Gao, A. Rzewski, H. Sun, P. D. Robbins, and A. Gambotto. UpGene: Application of a Web-Based DNA Codon Optimization Algorithm. *Biotechnology progress*, 20(2):443–448, 2004.

[65] P. Gaspar. *Gene optimization for heterologous expression.* PhD thesis, University of Aveiro, 2010.

[66] P. Gaspar, P. Lopes, J. Oliveira, R. Santos, R. Dalgleish, and J. L. Oliveira. Variobox: automatic detection and annotation of human genetic variants. *Human mutation*, 35(2):202–7, Feb. 2014.

[67] P. Gaspar and J. Luis Oliveira. Advantages of a Pareto-Based Genetic Algorithm to Solve the Gene Synthetic Design Problem. *Current Bioinformatics*, 7(3):304–309, Aug. 2012.

[68] P. Gaspar, G. Moura, M. a. S. Santos, and J. L. Oliveira. mRNA secondary structure optimization using a correlated stem-loop prediction. *Nucleic acids research*, 41:e73, 2013.

[69] P. Gaspar, J. L. Oliveira, J. Frommlet, M. a. S. Santos, and G. Moura. EuGene: maximizing synthetic gene design for heterologous expression. *Bioinformatics (Oxford, England)*, 28:2683–4, 2012.

[70] S. Geman, E. Bienenstock, and R. Doursat. Neural Networks and the Bias Variance Dilemma. *Neural Computation*, 4:1–58, 1992.

[71] G. S. Ginsburg and J. J. McCarthy. Personalized medicine: revolutionizing drug discovery and patient care. *Trends in Biotechnology*, 19(12):491–496, 2001.

[72] M. Gouy and C. Gautier. Codon usage in bacteria: correlation with gene expressivity. *Nucleic Acids Research*, 10(22):7055, 1982.

[73] A. Grote, K. Hiller, M. Scheer, R. Münch, B. Nörtemann, D. C. Hempel, and D. Jahn. JCat: a novel tool to adapt codon usage of a target gene to its potential expression host. *Nucleic acids research*, 33:W526–W531, 2005.

[74] G. A. Gutman and G. W. Hatfield. Nonrandom utilization of codon pairs in Escherichia coli. *Proceedings of the National Academy of Sciences of the United States of America*, 86:3699–3703, 1989.

[75] B. Hajek. Cooling Schedules for Optimal Annealing, 1988.

[76] M. N. Hall, J. Gabay, M. Débarbouillé, and M. Schwartz. A role for mRNA secondary structure in the control of translation initiation. *Nature*, 295:616–618, 1982.

[77] A. Hamosh, A. F. Scott, J. S. Amberger, C. A. Bocchini, and V. A. McKusick. Online Mendelian Inheritance in Man (OMIM), a knowledgebase of human genes and genetic disorders. *Nucleic acids research*, 33:D514–D517, 2005.

[78] S. Henikoff and J. G. Henikoff. Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Sciences of the United States of America*, 89:10915–10919, 1992.

[79] H. Herzel, O. Weiss, and E. N. Trifonov. 10-11 bp periodicities in complete genomes reflect protein structure and DNA folding. *Bioinformatics*, 15(3):187–193, 1999.

[80] I. L. Hofacker. Vienna RNA secondary structure server. *Nucleic acids research*, 31(13):3429–3431, 2003.

[81] D. M. Hoover and J. Lubkowski. DNAWorks: an automated method for designing oligonucleotides for PCR-based gene synthesis. *Nucleic acids research*, 30(10):e43, 2002.

[82] H. Hu, C. D. Huff, B. Moore, S. Flygare, M. G. Reese, and M. Yandell. VAAST 2.0: improved variant classification and disease-gene identification using a conservation-controlled amino acid substitution matrix. *Genetic epidemiology*, 37:622–34, 2013.

[83] T. Ikemura. Correlation between the abundance of yeast transfer RNAs and the occurrence of the respective codons in protein genes. Differences in synonymous codon choice patterns of yeast and Escherichia coli with reference to the abundance of isoaccepting transfer R. *Journal of molecular biology*, 158(4):573–597, 1982.

[84] T. Ikemura. Codon usage and tRNA content in unicellular and multicellular organisms. *Molecular biology and evolution*, 2:13–34, 1985.

[85] P. K. Ingvarsson. Gene expression and protein length influence codon usage and rates of sequence evolution in Populus tremula. *Molecular biology and evolution*, 24:836–844, 2007.

[86] S. Itzkovitz and U. Alon. The genetic code is nearly optimal for allowing additional information within protein-coding sequences. *Genome research*, 17(4):405–12, Apr. 2007.

[87] H. Jin, Q. Zhao, E. I. Gonzalez de Valdivia, D. H. Ardell, M. Stenström, L. A. Isaksson, and E. I. G. de Valdivia. Influences on gene expression in vivo by a Shine–Dalgarno sequence. *Molecular microbiology*, 60(2):480–492, 2006.

[88] M. Johnson, I. Zaretskaya, Y. Raytselis, Y. Merezhuk, S. McGinnis, and T. L. Madden. NCBI BLAST: a better web interface. *Nucleic acids research*, 36(suppl 2):W5, 2008.

[89] D. T. Jones. Protein secondary structure prediction based on position-specific scoring matrices. *Journal of molecular biology*, 292(2):195–202, Sept. 1999.

[90] G. F. Joyce. Evolution in an RNA world. *Cold Spring Harbor symposia on quantitative biology*, 74:17–23, 2009.

[91] P. Kadlec and Z. Raida. Comparison of novel multi-objective self organizing migrating algorithm with conventional methods. In *Proceedings of 21st International Conference, Radioelektronika 2011*, pages 97–100, 2011.

[92] H. J. Kim and S. J. Lee. Optimizing the secondary structure of human papillomavirus type 16 L1 mRNA enhances L1 protein expression in Saccharomyces cerevisiae. *Journal of biotechnology*, 150(1):31–36, 2010.

[93] M. Kircher, D. M. Witten, P. Jain, B. J. O'Roak, G. M. Cooper, and J. Shendure. A general framework for estimating the relative pathogenicity of human genetic variants. *Nature genetics*, pages 1–8, 2014.

[94] S. Kirkpatrick, C. D. Gelatt Jr, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671, 1983.

[95] M. Kirwan, A. J. Walne, V. Plagnol, M. Velangi, A. Ho, U. Hossain, T. Vulliamy, and I. Dokal. Exome sequencing identifies autosomal-dominant SRP72 mutations associated with familial aplasia and myelodysplasia. *American Journal of Human Genetics*, 90:888–892, 2012.

[96] R. D. Knight, S. J. Freeland, and L. F. Landweber. A simple model based on mutation and selection explains trends in codon and amino-acid usage and GC composition within and across genomes. *Genome biology*, 2:RESEARCH0010, 2001.

[97] J. D. Knowles and D. W. Corne. Approximating the nondominated front using the pareto archived evolution strategy. *Evolutionary computation*, 8(2):149–172, 2000.

[98] B. Knudsen and J. Hein. Pfold: RNA secondary structure prediction using stochastic context-free grammars. *Nucleic acids research*, 31(13):3423–3428, 2003.

[99] Y. Kojima, K.-Q. Xin, T. Ooki, K. Hamajima, T. Oikawa, K. Shinoda, T. Ozaki, Y. Hoshino, N. Jounai, M. Nakazawa, and Others. Adjuvant effect of multi-CpG motifs on an HIV-1 DNA vaccine. *Vaccine*, 20(23):2857–2865, 2002.

[100] A. a. Komar. A pause for thought along the co-translational folding pathway. *Trends in biochemical sciences*, 34(1):16–24, Jan. 2009.

[101] A. A. Komar, T. Lesnik, and C. Reiss. Synonymous codon substitutions affect ribosome traffic and protein folding during in vitro translation. *FEBS Letters*, 462:387–391, 1999.

[102] M. Kozak. Influences of mRNA secondary structure on initiation by eukaryotic ribosomes. *Proceedings of the National Academy of Sciences of the United States of America*, 83(9):2850, 1986.

[103] M. Kozak. An analysis of 5'-noncoding sequences from 699 vertebrate messenger RNAs. *Nucleic acids research*, 15:8125–8148, 1987.

[104] M. Kozak. Initiation of translation in prokaryotes and eukaryotes. *Gene*, 234(2):187–208, 1999.

[105] G. Kudla, A. W. Murray, D. Tollervey, and J. B. Plotkin. Coding-sequence determinants of gene expression in Escherichia coli. *Science*, 324:255–258, 2009.

[106] J. Kyte and R. F. Doolittle. A simple method for displaying the hydropathic character of a protein. *Journal of molecular biology*, 157:105–132, 1982.

[107] S. Levy, G. Sutton, P. C. Ng, L. Feuk, A. L. Halpern, B. P. Walenz, N. Axelrod, J. Huang, E. F. Kirkness, G. Denisov, Y. Lin, J. R. MacDonald, A. W. C. Pang, M. Shago, T. B. Stockwell, A. Tsiamouri, V. Bafna, V. Bansal, S. A. Kravitz, D. A. Busam, K. Y. Beeson, T. C. McIntosh, K. A. Remington, J. F. Abril, J. Gill, J. Borman, Y. H. Rogers, M. E. Frazier, S. W. Scherer, R. L. Strausberg, and J. C. Venter. The diploid genome sequence of an individual human. *PLoS Biology*, 5:2113–2144, 2007.

[108] B. Li, V. G. Krishnan, M. E. Mort, F. Xin, K. K. Kamati, D. N. Cooper, S. D. Mooney, and P. Radivojac. Automated inference of molecular mechanisms of disease from amino acid substitutions. *Bioinformatics (Oxford, England)*, 25:2744–2750, 2009.

[109] Z. Li, X. Liu, J. Wen, Y. Xu, X. Zhao, X. Li, L. Liu, and X. Zhang. DRUMS: a human disease related unique gene mutation search engine. *Human mutation*, 32(10):E2259–E2265, 2011.

[110] P. Lopes, R. Dalgleish, and J. L. Oliveira. WAVe: web analysis of the variome. *Human mutation*, 32(7):729–734, 2011.

[111] R. Lorenz, S. H. Bernhart, C. H. zu Siederdissen, H. Tafer, C. Flamm, P. F. Stadler, and I. L. Hofacker. ViennaRNA Package 2.0. *Algorithms for Molecular Biology*, 6(1):26, 2011.

[112] D. Lorimer, A. Raymond, J. Walchli, M. Mixon, A. Barrow, E. Wallace, R. Grice, A. Burgin, and L. Stewart. Gene Composer: database software for protein construct design, codon engineering, and gene synthesis. *BMC biotechnology*, 9(1):36, 2009.

[113] D. J. Lynn, G. A. C. Singer, and D. A. Hickey. Synonymous codon usage is subject to selection in thermophilic bacteria. *Nucleic acids research*, 30:4272–4277, 2002.

[114] C. D. Manning and H. Schütze. *Foundations of Natural Language Processing*. 2000.

[115] T. A. Manolio and F. S. Collins. The HapMap and genome-wide association studies in diagnosis and therapy. *Annual review of medicine*, 60:443–456, 2009.

[116] R. Marler and J. Arora. Survey of multi-objective optimization methods for engineering, 2004.

[117] V. Mathura and D. Kolippakkam. APDbase: Amino acid physicochemical properties database. *Bioinformation*, pages 2–4, 2005.

[118] B. W. Matthews. Comparison of the predicted and observed secondary structure of T4 phage lysozyme. *Biochimica et biophysica acta*, 405:442–451, 1975.

[119] L. J. McGuffin, K. Bryson, and D. T. Jones. The PSIPRED protein structure prediction server. *Bioinformatics (Oxford, England)*, 16:404–405, 2000.

[120] L. J. McGuffin, K. Bryson, and D. T. Jones. The PSIPRED protein structure prediction server. *Bioinformatics (Oxford, England)*, 16(4):404–5, Apr. 2000.

[121] A. Messac. Physical programming - Effective optimization for computational design, 1996.

[122] A. Messac. Aggregate objective functions and pareto frontiers: Required relationships and practical implications. *Optimization and Engineering Journal*, 1:171–188, 2000.

[123] M. Mort, T. Sterne-Weiler, B. Li, E. V. Ball, D. N. Cooper, P. Radivojac, J. R. Sanford, and S. D. Mooney. MutPred Splice: machine learning-based prediction of exonic variants that disrupt splicing. *Genome Biology*, 15(1):R19, 2014.

[124] G. Moura, M. Pinheiro, J. Arrais, A. C. Gomes, L. Carreto, A. Freitas, J. L. Oliveira, and M. a. S. Santos. Large scale comparative codon-pair context analysis unveils general rules that fine-tune evolution of mRNA primary structure. *PLoS One*, 2(9):e847, Jan. 2007.

[125] G. Moura, M. Pinheiro, R. Silva, I. Miranda, V. Afreixo, G. Dias, A. Freitas, J. Oliveira, and M. Santos. Comparative context analysis of codon pairs on an ORFeome scale. *Genome Biology*, 6(3):R28, 2005.

[126] M. W. Nachman and S. L. Crowell. Estimate of the mutation rate per nucleotide in humans. *Genetics*, 156:297–304, 2000.

[127] N. Naidoo, Y. Pawitan, R. Soong, D. N. Cooper, and C.-S. Ku. Human genetics and genomics a decade after the release of the draft sequence of the human genome, 2011.

[128] P. C. Ng and S. Henikoff. Predicting deleterious amino acid substitutions. *Genome research*, 11:863–874, 2001.

[129] P. C. Ng and S. Henikoff. SIFT: Predicting amino acid changes that affect protein function. *Nucleic acids research*, 31:3812–3814, 2003.

[130] P. C. P. Ng and S. Henikoff. Predicting the effects of amino acid substitutions on protein function. *Annual review of genomics and human genetics*, 7:61–80, Jan. 2006.

[131] S. B. Ng, A. W. Bigham, K. J. Buckingham, M. C. Hannibal, M. J. McMillin, H. I. Gildersleeve, A. E. Beck, H. K. Tabor, G. M. Cooper, H. C. Mefford, C. Lee, E. H. Turner, J. D. Smith, M. J. Rieder, K.-I. Yoshiura, N. Matsumoto, T. Ohta, N. Niikawa, D. A. Nickerson, M. J. Bamshad, and J. Shendure. Exome sequencing identifies MLL2 mutations as a cause of Kabuki syndrome. *Nature genetics*, 42:790–793, 2010.

[132] S. B. Ng, K. J. Buckingham, C. Lee, A. W. Bigham, H. K. Tabor, K. M. Dent, C. D. Huff, P. T. Shannon, E. W. Jabs, D. A. Nickerson, J. Shendure, and M. J. Bamshad. Exome sequencing identifies the cause of a mendelian disorder. *Nature genetics*, 42:30–35, 2010.

[133] M. Nirenberg, T. Caskey, R. Marshall, R. Brimacombe, D. Kellogg, B. Doctor, D. Hatfield, J. Levin, F. Rottman, S. Pestka, and Others. The RNA code and

protein synthesis. In *Cold Spring Harbor Symposia on Quantitative Biology*, volume 31, pages 11–24. Cold Spring Harbor Laboratory Press, 1966.

[134] R. Nussinov and A. B. Jacobson. Fast algorithm for predicting the secondary structure of single-stranded RNA. *Proceedings of the National Academy of Sciences*, 77(11):6309, 1980.

[135] H. Ogata, S. Goto, K. Sato, W. Fujibuchi, H. Bono, and M. Kanehisa. KEGG: Kyoto encyclopedia of genes and genomes, 1999.

[136] M. Ojala and G. Garriga. Permutation tests for studying classifier performance. *The Journal of Machine Learning Research*, 11:1833–1863, 2010.

[137] A. Olatubosun, J. Väliaho, J. Härkönen, J. Thusberg, and M. Vihinen. PON-P: integrated predictor for pathogenicity of missense variants. *Human mutation*, 33:1166–74, 2012.

[138] J. N. Onuchic and P. G. Wolynes. Theory of protein folding. *Current opinion in structural biology*, 14(1):70–5, Feb. 2004.

[139] R. A. Pagon and T. L. Trotter. Genetic testing: when to test and when to refer. *Paediatrics and Child Health*, 17:367–370, 2007.

[140] S. Pechmann and J. Frydman. Evolutionary conservation of codon optimality reveals hidden signatures of cotranslational folding. *Nature structural & molecular biology*, 20(2):237–43, Feb. 2013.

[141] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2012.

[142] C. Pesquita, D. Faria, A. O. Falc??o, P. Lord, and F. M. Couto. Semantic similarity in biomedical ontologies, 2009.

[143] T. D. Pham, J. O'Connell, and D. I. Crane. Constrained codon optimization by dynamic programming. In *Intelligent Multimedia, Video and Speech Processing, 2004. Proceedings of 2004 International Symposium on*, pages 153–156. IEEE, 2004.

[144] V. Phan, S. Saha, A. Pandey, and W. Tit-Yee. Synthetic Gene Design with a Large Number of Hidden Stop Codons. *IEEE International Conference on Bioinformatics and Biomedicine, 2008. BIBM'08*, pages 141–146, 2008.

[145] R. M. Plenge, M. Seielstad, L. Padyukov, A. T. Lee, E. F. Remmers, B. Ding, A. Liew, H. Khalili, A. Chandrasekaran, L. R. L. Davies, W. Li, A. K. S. Tan, C. Bonnard, R. T. H. Ong, A. Thalamuthu, S. Pettersson, C. Liu, C. Tian, W. V. Chen, J. P. Carulli, E. M. Beckman, D. Altshuler, L. Alfredsson, L. A. Criswell, C. I. Amos, M. F. Seldin, D. L. Kastner, L. Klareskog, and P. K. Gregersen. TRAF1-C5 as a risk locus for rheumatoid arthritis–a genomewide study. *The New England journal of medicine*, 357:1199–1209, 2007.

[146] S. E. Plon, D. M. Eccles, D. Easton, W. D. Foulkes, M. Genuardi, M. S. Greenblatt, F. B. L. Hogervorst, N. Hoogerbrugge, A. B. Spurdle, and S. V. Tavtigian. Sequence variant classification and reporting: recommendations for improving the interpretation of cancer susceptibility genetic test results. *Human mutation*, 29:1282–1291, 2008.

[147] J. B. Plotkin and G. Kudla. Synonymous but not the same: the causes and consequences of codon bias. *Nature reviews. Genetics*, 12:32–42, 2011.

[148] L. E. Post, G. D. Strycharz, M. Nomura, H. Lewis, and P. P. Dennis. Nucleotide sequence of the ribosomal protein gene cluster adjacent to the gene for RNA polymerase subunit beta in Escherichia coli. *Proceedings of the National Academy of Sciences of the United States of America*, 76:1697–1701, 1979.

[149] D. Powers. Evaluation: From Precision, Recall and F-measure to ROC, Informedness, Markedness & Correlation. *Journal of Machine Learning Technologies*, 2:37–63, 2011.

[150] P. Puigbò, E. Guzmán, A. Romeu, and S. Garcia-Vallvé. OPTIMIZER: a web server for optimizing the codon usage of DNA sequences. *Nucleic acids research*, 35(suppl 2):W126, 2007.

[151] A. Raymond, S. Lovell, D. Lorimer, J. Walchli, M. Mixon, E. Wallace, K. Thompkins, K. Archer, A. Burgin, and L. Stewart. Combined protein construct and synthetic gene engineering for heterologous protein expression and crystallization using Gene Composer. *BMC biotechnology*, 9:37, 2009.

[152] L. Ren, G. Gao, D. Zhao, M. Ding, J. Luo, and H. Deng. Developmental stage related patterns of codon usage and genomic GC content: searching for evolutionary fingerprints with models of stem cell differentiation. *Genome biology*, 8(3):R35, Jan. 2007.

[153] S. M. Richardson, P. W. Nunley, R. M. Yarrington, J. D. Boeke, and J. S. Bader. GeneDesign 3.0 is an updated synthetic biology toolkit. *Nucleic Acids Research*, 2010.

[154] H. Z. Ring, P. Y. Kwok, and R. G. H. Cotton. Human Variome Project: an international collaboration to catalogue human genetic variation. *Pharmacogenomics*, 7(7):969–972, 2006.

[155] C. A. Ross and M. A. Poirier. Protein aggregation and neurodegenerative disease. *Nature medicine*, 10 Suppl:S10–S17, 2004.

[156] H. M. W. Salim and A. R. O. Cavalcanti. Factors influencing codon usage bias in genomes. *Journal of the Brazilian Chemical Society*, 19:257–262, 2008.

[157] K. S. Sandhu, S. Pandey, S. Maiti, and B. Pillai. GASCO: genetic algorithm simulation for codon optimization. *In silico biology*, 8(2):187–192, 2008.

[158] P. Sasidharan Nair and M. Vihinen. VariBench: A Benchmark Database for Variations. *Human Mutation*, 34:42–49, 2013.

[159] Z. E. Sauna and C. Kimchi-Sarfaty. Understanding the contribution of synonymous mutations to human disease. *Nature reviews. Genetics*, 12:683–691, 2011.

[160] Z. E. Sauna, C. Kimchi-Sarfaty, S. V. Ambudkar, and M. M. Gottesman. Silent polymorphisms speak: how they affect pharmacogenomics and the treatment of cancer. *Cancer research*, 67:9609–9612, 2007.

[161] D. F. Schwarz, O. Hädicke, J. Erdmann, A. Ziegler, D. Bayer, and S. Möller. SNPtoGO: characterizing SNPs by enriched GO terms. *Bioinformatics (Oxford, England)*, 24:146–148, 2008.

[162] J. M. Schwarz, C. Rödelsperger, M. Schuelke, and D. Seelow. MutationTaster evaluates disease-causing potential of sequence alterations. *Nature methods*, 7(8):575–6, Aug. 2010.

[163] H. Seligmann and D. D. Pollock. The ambush hypothesis: hidden stop codons prevent off-frame gene reading. *DNA and cell biology*, 23(10):701–705, Oct. 2004.

[164] D. J. Selkoe. Folding proteins in fatal ways. *Nature*, 426:900–904, 2003.

[165] S. a. Shabalina, N. a. Spiridonov, and A. Kashina. Sounds of silence: synonymous nucleotides as a key to biological regulation and complexity. *Nucleic acids research*, 41:2073–94, 2013.

[166] P. M. Sharp and W. H. Li. The codon adaptation index - a measure of directional synonymous codon usage bias, and its potential applications. *Nucleic acids research*, 15(3):1281, 1987.

[167] P. M. Sharp, T. M. F. Tuohy, and K. R. Mosurski. Codon usage in yeast: cluster analysis clearly differentiates highly and lowly expressed genes. *Nucleic acids research*, 14(13):5125, 1986.

[168] H. a. Shihab, J. Gough, D. N. Cooper, P. D. Stenson, G. L. a. Barker, K. J. Edwards, I. N. M. Day, and T. R. Gaunt. Predicting the Functional, Molecular, and Phenotypic Consequences of Amino Acid Substitutions using Hidden Markov Models. *Human mutation*, 2012.

[169] H. a. Shihab, J. Gough, D. N. Cooper, P. D. Stenson, G. L. a. Barker, K. J. Edwards, I. N. M. Day, and T. R. Gaunt. Predicting the functional, molecular, and phenotypic consequences of amino acid substitutions using hidden Markov models. *Human mutation*, 34:57–65, 2013.

[170] J. Shine and L. Dalgarno. The 3'-terminal sequence of Escherichia coli 16S ribosomal RNA: complementarity to nonsense triplets and ribosome binding sites. *Proceedings of the National Academy of Sciences*, 71(4):1342, 1974.

[171] D. Smith and M. Yarus. tRNA-tRNA interactions within cellular ribosomes. *Proceedings of the National Academy of Sciences of the United States of America*, 86:4397–4401, 1989.

[172] J. Starmer, a. Stomp, M. Vouk, and D. Bitzer. Predicting Shine-Dalgarno sequence locations exposes genome annotation errors. *PLoS computational biology*, 2(5):e57, May 2006.

[173] P. D. Stenson, M. Mort, E. V. Ball, K. Howells, A. D. Phillips, N. S. Thomas, and D. N. Cooper. The Human Gene Mutation Database: 2008 update. *Genome medicine*, 1:13, 2009.

[174] S. M. Studer and S. Joseph. Unfolding of mRNA Secondary Structure by the Bacterial Translation Initiation Complex. *Molecular Cell*, 22:105–115, 2006.

[175] S. M. Studer and S. Joseph. Unfolding of mRNA Secondary Structure by the Bacterial Translation Initiation Complex. *Molecular Cell*, 22:105–115, 2006.

[176] Y. Sun, S. H. Ye, and H. W. Lu. Study of RNA Secondary Structure Prediction Algorithms. *Advanced Materials Research*, 393:955–960, 2012.

[177] S. Sunyaev, V. Ramensky, I. Koch, W. Lathe, A. S. Kondrashov, and P. Bork. Prediction of deleterious human alleles. *Human molecular genetics*, 10:591–597, 2001.

[178] K. Tamura, J. Dudley, M. Nei, and S. Kumar. MEGA4: Molecular Evolutionary Genetics Analysis (MEGA) software version 4.0. *Molecular biology and evolution*, 24(8):1596–1599, 2007.

[179] A. Tats, T. Tenson, and M. Remm. Preferred and avoided codon pairs in three domains of life. *BMC genomics*, 9:463, 2008.

[180] P. D. Thomas, M. J. Campbell, A. Kejariwal, H. Mi, B. Karlak, R. Daverman, K. Diemer, A. Muruganujan, and A. Narechania. PANTHER: a library of protein families and subfamilies indexed by function. *Genome research*, 13:2129–2141, 2003.

[181] J. Thusberg, A. Olatubosun, and M. Vihinen. Performance of mutation pathogenicity prediction methods on missense variants. *Human mutation*, 32(4):358–68, Apr. 2011.

[182] D. Ting, G. Wang, M. Shapovalov, R. Mitra, M. I. Jordan, and R. L. Dunbrack. Neighbor-dependent Ramachandran probability distributions of amino acids developed from a hierarchical Dirichlet process model. *PLoS computational biology*, 6(4):e1000763, Apr. 2010.

[183] T. Tuller, A. Carmi, K. Vestsigian, S. Navon, Y. Dorfan, J. Zaborske, T. Pan, O. Dahan, I. Furman, and Y. Pilpel. An evolutionarily conserved mechanism for controlling the efficiency of protein translation. *Cell*, 141(2):344–354, 2010.

[184] B. W. van Bon, C. Gilissen, D. K. Grange, R. C. Hennekam, H. Kayserili, H. Engels, H. Reutter, J. R. Ostergaard, E. Morava, K. Tsiakas, B. Isidor, M. Le Merrer, M. Eser, N. Wieskamp, P. de Vries, M. Steehouwer, J. A. Veltman, S. P. Robertson, H. G. Brunner, B. B. de Vries, and A. Hoischen. Cantu syndrome is caused by mutations in ABCC9. *Am J Hum Genet*, 90:1094–1101, 2012.

[185] D. A. Van Veldhuizen and G. B. Lamont. Evolutionary Computation and Convergence to a Pareto Front. In *Late Breaking Papers at the Genetic Programming 1998 Conference*, pages 221–228, 1998.

[186] M. Vihinen. How to evaluate performance of prediction methods? Measures and their interpretation in variation effect analysis, 2012.

[187] A. Villalobos, J. E. Ness, C. Gustafsson, J. Minshull, and S. Govindarajan. Gene Designer: a synthetic biology tool for constructing artificial DNA segments. *BMC bioinformatics*, 7(1):285, Jan. 2006.

[188] R. Vogelbacher, D. Angulo, and S. Koide. Sequence optimization for synthetic genes using a genetic algorithm. In *Proceedings of the Midwest Software Engineering Conference/DePaul CTI Research Symposium*, volume 1. Citeseer, 2006.

[189] G.-S. Wang and T. A. Cooper. Splicing in disease: disruption of the splicing code and the decoding machinery. *Nature reviews. Genetics*, 8:749–761, 2007.

[190] K. Wang, M. Li, and H. Hakonarson. ANNOVAR: functional annotation of genetic variants from high-throughput sequencing data. *Nucleic acids research*, 38(16):e164–e164, 2010.

[191] Z. Wang and C. Burge. Splicing regulation: from a parts list of regulatory elements to an integrated splicing code. *Rna*, pages 802–813, 2008.

[192] Z. Wang and J. Moult. SNPs, protein structure, and disease. *Human Mutation*, 17:263–270, 2001.

[193] J. Watson and F. Crick. Molecular structure of nucleic acids. *Nature*, 171:737–738, 1953.

[194] T. Weise. Global Optimization Algorithms - Theory and Application. 1, 2009.

[195] M. Welch, S. Govindarajan, J. E. Ness, A. Villalobos, A. Gurney, J. Minshull, and C. Gustafsson. Design parameters to control synthetic gene expression in Escherichia coli. *PloS one*, 4(9):e7002, Jan. 2009.

[196] M. Welch, A. Villalobos, C. Gustafsson, and J. Minshull. You're one in a googol: optimizing genes for protein expression. *Journal of The Royal Society Interface*, 6(Suppl 4):S467, 2009.

[197] I. Weygand-Durasevic and M. Ibba. New roles for codon usage. *Science(Washington)*, 329(September), 2010.

[198] D. H. Wolpert and W. G. Macready. No free lunch theorems for search. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997.

[199] F. Wright. The effective number of codons used in a gene. *Gene*, 87(1):23–29, 1990.

[200] M. Yarus and L. S. Folley. Sense codons are found in specific contexts. *Journal of molecular biology*, 182:529–540, 1985.

[201] G. Zhang, M. Hubalewska, and Z. Ignatova. Transient ribosomal attenuation coordinates protein synthesis and co-translational folding. *Nature structural & molecular biology*, 16(3):274–280, 2009.

[202] W. Zhang, W. Xiao, H. Wei, J. Zhang, and Z. Tian. mRNA secondary structure at start AUG codon is a key limiting factor for human protein expression in Escherichia coli. *Biochemical and biophysical research communications*, 349(1):69–78, 2006.

[203] Y. Zhang, J. Zhang, H. Hara, I. Kato, and M. Inouye. Insights into the mRNA cleavage mechanism by MazF, an mRNA interferase. *The Journal of biological chemistry*, 280:3143–3150, 2005.

[204] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation*, 3:257–271, 1999.

[205] M. Zuker. Mfold web server for nucleic acid folding and hybridization prediction. *Nucleic acids research*, 31(13):3406–3415, 2003.

[206] M. Zuker and P. Stiegler. Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information. *Nucleic acids research*, 9(1):133, 1981.

A | GENE EVALUATION ALGORITHMS

A.1 CODON PAIR BIAS EVALUATION

Algorithm A.1 Gene CPB evaluation

$N \leftarrow$ length of gene in number of codons
for $i = 1$ to $N - 1$ **do**
  $codon_a \leftarrow$ codon at position i
  $codon_b \leftarrow$ codon at position $i + 1$
  CPS $= ln\left(\frac{F_{a,b}}{\frac{F_a F_b}{F_x F_y} F_{x,y}}\right)$ ▷ As explained in Chapter 2
  $accumulator \leftarrow accumulator + CPS$
end for
return $\frac{accumulator}{N}$

An adaptation of the same algorithm can be used to also evaluate for out-of-frame stop codons. The change is made by replacing line 5 with a stop verification.

A.2 CODON USAGE EVALUATION

Algorithm A.2 Gene RSCU evaluation

$N \leftarrow$ length of gene in number of codons
for $i = 1$ to N **do**
  $RSCU_{codoni} \leftarrow \frac{F_{i,j}}{\frac{1}{n_i}\sum_{k=1}^{n_j} F_{k,j}}$ ▷ As explained in Chapter 2
  $accumulator \leftarrow accumulator + RSCU_{codoni}$
end for
return $\frac{accumulator}{N}$

Algorithm A.3 Gene CAI evaluation

$N \leftarrow$ length of gene in number of codons
$accumulator \leftarrow 1$
for $i = 1$ to N **do**
  $RSCU_{codon} \leftarrow \frac{F_{i,j}}{\frac{1}{n_i}\sum_{k=1}^{n_j} F_{k,j}}$
  $w_{codon} \leftarrow \frac{RSCU_{codon}}{RSCU_{imax}}$
  $accumulator \leftarrow accumulator \times RSCU_{codon}$
end for
return $accumulator^{1/N}$

To account for the availability of tRNAs, the tAI measure can be applied by replacing the calculation of w_{codon} by the formula describe in equation 3 of chapter 2.

A.3 GENE INTERACTION SITES

Algorithm A.4 Gene interactions evaluation

$siteList \leftarrow$ list of known interaction sites
$prefixTree \leftarrow$ build trie from $siteList$ ▷ using the Aho-Corasick algorithm
$N \leftarrow$ length of gene in number of nucleotides
$accumulator \leftarrow 0$
for $i = 1$ to N **do**
  $nuc \leftarrow$ nucleotide in position i
  $score \leftarrow$ get score for nuc from prefixTree
  $accumulator \leftarrow accumulator + score$
end for
return $\frac{accumulator}{N}$

An adaptation of the algorithm can be performed in order to evaluate similarity considering the possibility for unexact matches.

A.4 AMINO ACID STARVATION

Algorithm A.5 Gene AA overuse evaluation

$N \leftarrow$ length of gene in number of codons
$Tlist \leftarrow$ list of tRNAs for each amino acid
$accumulators \leftarrow$ array of zeros with the size of Tlist
for $i = 1$ to N **do**
    $T \leftarrow Tlist$[amino acid in position i]
    $accumulators[T] \leftarrow accumulators[T] + 1$
end for
return $\sum_j accumulators[j]$

A.5 CODON CORRELATION EFFECT

Algorithm A.6 Codon correlation effect

$codonFraction \leftarrow 20$
$accumulator \leftarrow 0$
for $i = 1$ to $codonFraction$ **do**
    $codonlist \leftarrow codonlist + codon_i$
end for
for $i = codonFraction + 1$ to N **do**
    if $codon_i$ in $codonList$ **then**
        $accumulator \leftarrow accumulator + 1$
    end if
end for
return $accumulator/(N - codonFraction)$

A.6 GENE REPEATS

Algorithm A.7 Gene nucleotide repeats evaluation

$N \leftarrow$ length of gene in number of nucleotides
$previous \leftarrow$ nucleotide in position 1
$minimumRepeats \leftarrow 5$
$counter \leftarrow 0$
$accumulator \leftarrow 0$
for $i = 2$ to N **do**
  $current \leftarrow$ nucleotide in position i
  if $previous = current$ **then**
    $counter \leftarrow counter + 1$
  else
    if $counter \geq minimumRepeats$ **then**
      $accumulator \leftarrow accumulator + counter$
    end if
    $counter \leftarrow 0$
  end if
  $previous \leftarrow current$
end for
return $\frac{accumulator}{N}$

A.7 GENE RAMP EFFECT

Algorithm A.8 Gene ramp effect evaluation

$codonFraction \leftarrow 20$
$accumulator \leftarrow 0$
for $i = 1$ to $codonFraction$ **do**
  $RSCU_{codoni} \leftarrow \frac{F_{i,j}}{\frac{1}{n_i}\sum_{k=1}^{n_j} F_{k,j}}$
  $accumulator \leftarrow accumulator + RSCU_{codoni}$
end for
return $accumulator/(codonFraction)$

A.8 GENE GUANINE AND CYTOSINE CONTENT

Algorithm A.9 Gene GC% evaluation

$N \leftarrow$ length of gene in number of codons
$accumulator \leftarrow 0$
for $i = 1$ to N **do**
    $GC_{codoni} \leftarrow$ amount of guanine and cytosine in codon
    $accumulator \leftarrow accumulator + GC_{codoni}$
end for
return $\frac{accumulator}{N}$

B | MRNA SECONDARY STRUCTURE OPTIMIZATION TESTS

In order to compare our aproach with other methods, we selected 3 genes from *Aquifex aeolicus*, with different lengths, and performed the codon optimization using different known RNA structure predictors. In each method, the final evaluation is made with both the method and RNAfold to create a comparable and reliable basis between the methods. The gain in using each method instead of our pseudo-energy optimization (Gain over PE), and the time for each optimization, are also shown. An extensive test cannot be performed given the large time complexity of some algorithms.

Genome	# of codons	GC Content	Gene	Wild Type	Optimization guided by *Pseudo-Energy (PE)*				
				RNAfold Energy	Method Energy	**RNAFold Energy**	**RNAFold Improvement**	**Time (seconds)**	**Gain over PE**
Aquifex aeolicus	105	46%	rpsJ	-76,32	-27,8	-31,2	59%	4	-
	193	46%	hisB	-172,03	-91,5	-92,3	46%	10	-
	269	44%	cysQ	-244,40	-131,0	-111,7	54%	18	-
						Average:	**53%**	**10**	

Genome	# of codons	GC Content	Gene	Optimization guided by *RNAFold*				
				Method Energy	**RNAFold Energy**	**RNAFold Improvement**	**Time (seconds)**	**Gain over PE**
Aquifex aeolicus	105	46%	rpsJ	-	-25,6	66%	416	7%
	193	46%	hisB	-	-88,4	49%	1836	2%
	269	44%	cysQ	-	-119,5	51%	3405	-3%
					Average:	**55%**	**1886**	2%

Genome	# of codons	GC Content	Gene	Optimization guided by *UNAFold* Method Energy	RNAFold Energy	RNAFold Improvement	Time (seconds)	Gain over PE
Aquifex aeolicus	105	46%	rpsJ	-21,7	-37,5	51%	1110	-8%
	193	46%	hisB	-65,1	-93,4	46%	5063	-1%
	269	44%	cysQ	-78,8	-127,4	48%	11018	-6%
					Average:	**48%**	**5730**	-5%

Genome	# of codons	GC Content	Gene	Optimization guided by *PknotsRG* Method Energy	RNAFold Energy	RNAFold Improvement	Time (seconds)	Gain over PE
Aquifex aeolicus	105	46%	rpsJ	-27,8	-28,7	62%	45279	3%
	193	46%	hisB	-113,3	-116,1	33%	234761	-14%
	269	44%	cysQ	-140,4	-144,5	41%	510710	-13%
					Average:	**45%**	**263583**	-8%

Genome	# of codons	GC Content	Gene	Optimization guided by *RNAStructure* Method Energy	RNAFold Energy	RNAFold Improvement	Time (seconds)	Gain over PE
Aquifex aeolicus	105	46%	rpsJ	-20,1	-29,63	61%	26452	2%
	193	46%	hisB	-69,8	-96,2	44%	175316	-2%
	269	44%	cysQ	-91,3	-130,3	47%	374521	-8%
					Average:	**51%**	**192096**	-3%

Table B.1 Structure optimization with several methods. The first block represents the optimization resorting to our energy measurement algorithm (pseudo-energy). The wild type (original) energy is also shown, as measured by RNAfold. In subsequent blocks, the optimization is performed resorting to different RNA secondary structure predictors to perform evaluations: RNAfold, UNAFold, PknotsRG, and RNAStructure. For each of them, the energy of the final optimized gene is also measured with RNAfold, to establish a comparison basis. The "RNAFold improvement" indicates how much better the optimized gene is than the original gene. We also measure how much gain in improvement there is in comparison with our method. Only RNAFold achieves results better than ours in terms of optimization results. PknotsRG took the most time with an average of 3 days to optimize each gene.

C | GENE ONTOLOGY FEATURES

Term ID	Term Name
GO:0003824	catalytic activity
GO:0032502	developmental process
GO:0005215	transporter activity
GO:0044464	cell part
GO:0009987	cellular process
GO:0044421	extracellular region part
GO:0045735	nutrient reservoir activity
GO:0008152	metabolic process
GO:0005198	structural molecule activity
GO:0044423	virion part
GO:0032501	multicellular organismal process
GO:0031974	membrane-enclosed lumen
GO:0031386	protein tag
GO:0044699	single-organism process
GO:0045499	chemorepellent activity
GO:0005623	cell
GO:0030054	cell junction
GO:0009295	nucleoid
GO:0016530	metallochaperone activity
GO:0042056	chemoattractant activity
GO:0000003	reproduction
GO:0048511	rhythmic process
GO:0016015	morphogen activity
GO:0009055	electron carrier activity
GO:0023052	signaling
GO:0051234	establishment of localization
GO:0001071	nucleic acid binding transcription factor activity

GO:0022414	reproductive process
GO:0044456	synapse part
GO:0005488	binding
GO:0032991	macromolecular complex
GO:0005576	extracellular region
GO:0030234	enzyme regulator activity
GO:0040007	growth
GO:0071840	cellular component organization or biogenesis
GO:0016209	antioxidant activity
GO:0051179	localization
GO:0051704	multi-organism process
GO:0050896	response to stimulus
GO:0022610	biological adhesion
GO:0044420	extracellular matrix part
GO:0002376	immune system process
GO:0043226	organelle
GO:0016247	channel regulator activity
GO:0045202	synapse
GO:0065007	biological regulation
GO:0030545	receptor regulator activity
GO:0016020	membrane
GO:0040011	locomotion
GO:0060089	molecular transducer activity
GO:0000988	protein binding transcription factor activity
GO:0004872	receptor activity
GO:0031012	extracellular matrix
GO:0044422	organelle part
GO:0044425	membrane part
GO:0001906	cell killing
GO:0045182	translation regulator activity
GO:0019012	virion
GO:0036370	D-alanyl carrier activity
GO:0055044	symplast
GO:0097423	mitochondrion-associated adherens complex

Table C.1 Gene Ontology IDs and term names. All terms belong to the second level of the Gene Ontology graph.

D | FULL FEATURES LIST

List of features used in the machine learning environment to predict pathogenicity.

Amino Acid related	**Codon related**
AA affected secondary structure	codon GC content change
AA average mass	codon mutation position
AA hydropathy	codon percentile position
AA hydropathy change	codon RSCU change
AA isoelectric point	codon lost a stop
AA isoelectric point change	codon-next gained a stop
AA mass change	codon-next lost a stop
AA net charge	codon-prev gained a stop
AA net charge change	codon-prev lost a stop
AA pK1	codon pair score orig-next
AA pK1 change	codon pair score orig-next change
AA pK2	codon pair score prev-orig
AA pK2 change	codon pair score prev-orig change
AA pKa	codon pair score prev-to-next
AA pKa change	codon pair score prev-to-next change
AA polarity	is nonsense mutation
AA polarity change	is synonymous mutation
AA Ramacha Next Phi Diff	max repeats codon-next (mutated)
AA Ramacha Next Psi Diff	max repeats codon-next (wild-type)
AA Ramacha Prev Phi Diff	max repeats prev-codon (mutated)
AA Ramacha Prev Psi Diff	max repeats prev-codon (wild-type)
AA structure difference	windowed RSCU
AA van der Waals volume	windowed RSCU change percentage
AA van der Waals volume change	wild-type windowed number of stop codons
AA windowed averageMass change	wild-type codon GC content
AA windowed hydropathy change	wild-type codon RSCU
AA windowed isoelectric Pt	wild-type windowed GC content
AA windowed isoelectric Pt change	wild-type windowed num repeats
AA windowed net charge	wild-type windowed num repeats change

AA windowed net charge change	mutation position inside codon
AA windowed pK1	
AA windowed pK1 change	**Gene related**
AA windowed pK2	gene codon pair bias change
AA windowed pK2 change	gene RSCU change
AA windowed pKa	gene sec struct energy change (log diff)
AA windowed polarity	gene size
AA windowed polarity change	mRNA secondary struct energy
AA windowed van der Waals change	wild-type gene avg RSCU
windowed average mass	wild-type gene codon pair bias
windowed hydropathy	wild-type gene hidden stops ratio
windowed number of a-helix amino acids	wild-type gene nuc repeats ratio
windowed number of b-strand amino acids	wild-type windowed secondary structure (log)
windowed number of coil amino acids	windowed secondary structure change (log diff)
windowed pKa change	
windowed van der Waals volume	**Splicing regulation related**
	exonic splicing enhancer orig score
Gene Ontology related	exonic splicing enhancer score change
see list of terms in Appendix C	exonic splicing silencer orig score
	exonic splicing silencer score change
	distance of mutation to nearest exon edge (log)

Table D.1 Full list of features. AA stands for Amino Acid; windowed features were calculated in windows of 7 codons/amino acids centered at the mutation site; codon-next/prev refers to characteristics measured at the affected codon site and the following/preceding codon site.