THESE POUR OBTENIR LE GRADE DE DOCTEUR DE L'UNIVERSITE DE MONTPELLIER

En Mathématique appliquées et application des mathématiques

École doctorale : CBS2 – Sciences chimiques et Biologiques pour la santé

Unités de recherche : Institut de Génétique Moléculaire de Montpellier (IGMM – UMR 5535)
Institut Montpelliérain Alexandre Grothendieck (IMAG – UMR 5194)

Probing sequence-level instructions for gene expression

Présentée par May TAHA
Le 28 Novembre 2018

Sous la direction de Charles-Henri Lecellier
et Jean Michel Marin

Devant le jury composé de

Dr. Julien CHIQUET, Chargé de recherche, AgroParisTech/INRA MIA, Paris	Rapporteur
Dr. Mohamed ELATI, Professeur, Université de Lille	Rapporteur
Dr. Stéphane ROBIN, Directeur de recherche, AgroParisTech/INRA MIA, Paris	Examinateur
Dr. Nathalie VILLA-VIALANEIX, Chargé de recherche, Université de Toulouse 1	Examinateur
Dr. Charles-Henri LECELLIER, Chargé de recherche, CNRS, Montpellier	Directeur de thèse
Dr. Jean Michel MARIN, Professeur, Université Montpellier	Co-directeur de thèse

UNIVERSITÉ DE MONTPELLIER

It always seems impossible, until it's done.

– Nilson Mandela

In the honor of my father, hope you are as proud as I am.

Acknowledgements

After an intensive period of three and a half years, today is the day. Writing this note is the finishing touch of my dissertation. It has been a period of intense learning, not only on a scientific but also on a personal level. Writing this dissertation had a significant impact on me. I want to reflect on the people who have supported and helped me so much throughout this period.

Foremost, I would like to express my sincere gratitude to my directors, Charles-Henri Lecellier and Jean Michel Marin and my supervisors Laurent Bréhélin and Sophie Lébre. Charles, first thanks for the effort you made in explaining everything clearly and just, you helped in making biology fun for me. I am grateful for the encouragement, sound advice, and many good ideas, especially throughout my thesis-writing period. Your enthusiasm for science inspires me. More importantly, I am glad for the times you made me challenge my self. For you Jean-Michel, thank you for constant support, availability and constructive suggestions, which were determinant for the accomplishment of the work presented in this thesis. I admire your knowledge, your wisdom and your sense of humor, you made all our meeting very joyful and meaningful. Laurent and Sophie, this work would not be accomplished without your vast contributions. You invested your time in me and in the project, and you were here until the last sentence of this report was written. I cannot thank you enough for the hard work you made with me to accomplish this project and to finish a very well written and organized report. Finally, thank you all, especially Sophie, for the moral support when things were not that easy for me.

Beside my supervisors and co-supervisors, I would like to thank the rest of my thesis committee: Julien Chiquet and Mohamed Elati who did me the honor of reading this thesis

despite their busy schedule. Moreover, a special thank to Stéphane Robin and Nathalie Villa who accepted to be a part of this jury without hesitating. I would also like to thank Jacques Colinge who with Julien followed me from the beginning. Your remarks and propositions during my thesis committees helped me go forward and had a massive impact on this project. Finally, I would like to thank the école doctorale CBS2 for funding my thesis project for three years as well as for the continuous help in administrative procedures.

More generally, I would like to thank all the members of IGMM, especially the EBL and RBL, the members of IMAG especially the EPS team and all the members of the IBC group. Being in three different labs, with very different aspects, between biology, maths and its applications (statistics, informatics) helped me to widen my knowledge in different domains. I would like to thank François David Collin for all the help he gave with my project on deep learning. Thank you for the scientific discussions about the models as well as for always saving the day when it comes to the GPU and installations. I want also to thank every person with whom I had a scientific discussion, and for those who gave remarks and potential solutions during my presentations, every single idea counts. To Christophe and Raphel, thank you for the lovely time we shared in the office, and for the endless discussion about what is better R or Python. Note that now I feel confused after using both. To Fabienne, Mathias, Pierre, Etienne, Amani, and Baraa, it would not be the same without you. Thank you for making me laugh all the time, for the moral support especially during this last phase of writing and especially Mathias thanks for all the appropriate Arabic words you shared with me, you made me feel like home.

During these three years, it was not only science, but I was also fortunate to make long life friends. First of all, to the one whom I do not think that I would be here without her support, to you Chloé. I think you know, without saying, the considerable effect you made just being by my side. If I start to count why I am grateful to you, I need another PhD report, but I will try to summarize. You were and still huge moral support, we laughed and cried together, we complained about everything and drank martinis (this is why we are friends). I appreciate making me feel like home with your family and here a special appreciation to your mother who is a blessing. Finally, I am grateful just for everything and

especially for tasting all my crazy new receipts. Second, to Tiffany, my sweet little lovely girl, you were always there even when I was in the worst possible situation, you spread your love and support. Thank you for every message, every significant gift, for all the pandas in my room, thank you for being you. Adham, I do not know what to say, you have it all, my Lebanese partner in the team and in food at lunch. I love how smart and funny you are. I do not want to imagine the lab without you, without your social Mondays and almost weekend Fridays, without your jokes (always inappropriate) and your laugh. You are an extraordinary person, and I think you will be a very successful man in the future. I love you.

Besides the friends I made in the lab, I also have a second big Lebanese family, and I wish I do not forget anyone. For, Mazen, Nathalie, Costy, Jessica, Emiline, Bob, Johny, Joseph, Charbel, Mohamed (and antar for sure), Faten, Chadi, Hani, and Mahmoud in Montpellier, thank for making from Montpellier something like home. Every one of you knows how much I love and appreciate him and how much I am grateful for every piece of support you ever gave me. The Lebanese community does not stop in Montpellier, for those who are far by distance, are close by heart and mind, Ihsan, Hikmat, Josephine, Joanna, Sophie, Sally, and Hassan M.
Besides, even though it has been 4 years that I left, I still have terrific friends that supported me all over the road. For Mohamad H, Hussein A., Rana H., Said, Mariam and for Zeinab for the spelling corrections, a huge appreciation. Finally, to you, Idrisso, not only you support me infinitely, but you show me every day how much you are proud of me. You tolerated my bad mood and my growl for months while writing the report, you helped me in every single way to accomplish my work, and you always push me to be a better person. I, in my turn, am very proud you and very grateful that you are a huge part of my life. I wish you success in your life, and I am confident that you will accomplish all your dreams.

I thought about what to say to Yara for as long as I have been writing this thesis. It is hard to express what I would like you to know. I kept this particular paragraph for you just before my family because for me you are one of them. For three years, everyone thought that we are blood-related, or that we knew each other a long time ago. The truth is even though we have opposite personalities, we always understand and support each

other. Being thankful is not enough for you. We are a true friend, and a great support system, you yell at me when I act crazy or because I cry a lot, you eat very unhealthy food with me when I am sad, and finally for criticizing, every single day, my ingenuity of being a messy person. I want to tell you that I believe in you and I believe you are one of the most successful, organized, organized and organized women I ever met after my mother. Much Much love “azizati”.

Last but not least, my beloved family. This part was the hardest of writing. To my parents, who were by my side every single time, I can’t thank you enough for everything you do and for all the scarifies you make so, my siblings and I, are very successfully, independent and happy. Thank you for all the lessons you taught me about life, work, love, and family. You taught me to value everything, and for me, you are the best parents in the world. I know how proud you are and I promise to make you proud every single day. For my sister, brothers, and my sister in law you are the best support system in the world, my love for you is endless.

Contents

List of Figures

List of Algorithms

List of Tables

Objective and overview

Gene regulation is tightly controlled to ensure a wide variety of cell types and functions. These controls take place at different levels (transcriptional, post-transcriptional, ...) and are associated with distinct genomic regulatory regions (promoters, exons, introns, ...). An actual challenge is to understand how the gene regulation machinery works in each cell type and to identify the most important regulators. Several studies attempt to understand the regulatory mechanisms by modeling gene expression using epigenetic marks. Nonetheless, these approaches rely on experimental data which are limited to some samples, costly and time-consuming. Besides, the critical component of gene regulation based at the sequence level cannot be captured by these approaches. Our primary objective is to explain mRNA expression based only on DNA sequences features.

In this thesis, we assessed the ability of sequence-levels features to regulate gene expression in the different cancer type. To achieve this objective, we propose two statistical approaches able to explain gene expression in each patient sample using only DNA features. The first chapter is a background on the molecular biology, statistics and informatics fields introducing different terminologies and models of interest. It is also an opportunity to present a brief state of the art related to modeling gene expression and its controls, tackled in this dissertation. Each chapter is then dedicated to my contributions of the field of modeling gene expression based only on DNA sequence. In Chapter 2 we present an approach based on a Lasso penalized regression. Chapter 3 is an attempt to improve the model retained in Chapter 2. In Chapter 4, we provide a novel model to predict gene expression based on neural networks in different architecture: deep neural networks and convolution networks. The final chapter is a global discussion about all the results and limitations of our models as well as essential perspectives to tackle.

Chapter 1

State of the art

This chapter aims to describe the biological and statistical background of this thesis. First of all, we briefly introduce relevant aspects of molecular biology related to gene regulation in human cells in Section 1.1. In Section 1.2, we present statistical analyses and machine learning models used in this thesis. Finally, we review different biostatistics and bioinformatics studies about gene expression and its regulation in human and other organisms in Section 1.4.

1.1 Biological background

In this part, we introduce the different key aspects of gene regulation in the eukaryotic cell at the transcriptional and post-transcriptional levels. First, we describe different biological elements involved in gene regulation.

1.1.1 DNA and RNA

1.1.1.1 DNA and chromatin

Deoxyribonucleic acid (DNA) is a molecule that encodes genetic information. DNA contains the instructions that an organism needs to develop, live and reproduce. These instructions are found inside every cell and are passed down from parents to their children. Frederich Miescher first presented DNA in 1869 [MR71], but the importance of DNA re-

mained mainly unknown. In 1953, Watson-Crick, Wilkins, and Franklin ([WC+53],[WSW53], [FG53]) changed our understanding of biology by resolving the structure of DNA and realizing that it carries biological information.

DNA is mostly located in the cell nucleus. It is stored as a code of four nucleotides: adenine (A), guanine (G), cytosine (C), and thymine (T). Human DNA consists of about 3 billion bases [Ann08]. The order of the bases on the DNA (*i.e.* sequence) determines the genetic information as well as its function in the cell. DNA is formed of two antiparallel complementary strands with A being complementary of T and G complementary of C. Nucleotides are arranged in two long strands that form a spiral called a double helix defining the structure of the DNA. The senses of each strand are determined by the 5'phosphate of the first base and 3'OH ends of the last base. In human cells, DNA totals about 2 meters in length and is organized into compact structures called chromosomes. Each chromosome is made up of one DNA coiled tightly around proteins called histones that support its structure. The complexes between DNA and these proteins are called chromatin. The human genome contains 23 pairs of chromosomes. Figure 1.1 shows DNA structured in a chromosome.

Architecture of the chromatin

The development of Chromosome Conformation Capture technologies has led to the discovery of sets of physical interactions of DNA, between chromosomes (inter) or within the same chromosome (intra), known as chromatin domains.
First, Hi-C for High-throughput Chromosome Conformation Capture was introduced in 2009 by Lieberman-Aiden *& al.* [LAVBW+09] to explore the three-dimensional architecture across the entire genome. Using a special adaptation of the Chromosome Conformation Capture (3C), they were able to identify long-range interactions between pairs of loci (fixed position on the chromosome). In that same year, another mapping process was introduced to detect high order interactions, ChIA-PET [FLP+09] (Chromatin Interaction Analyses by Paired-End Tag sequencing) based on Chromatin Immunoprecipitation (ChIP). ChIA-PET identifies chromatin interactions between distal and proximal regula-

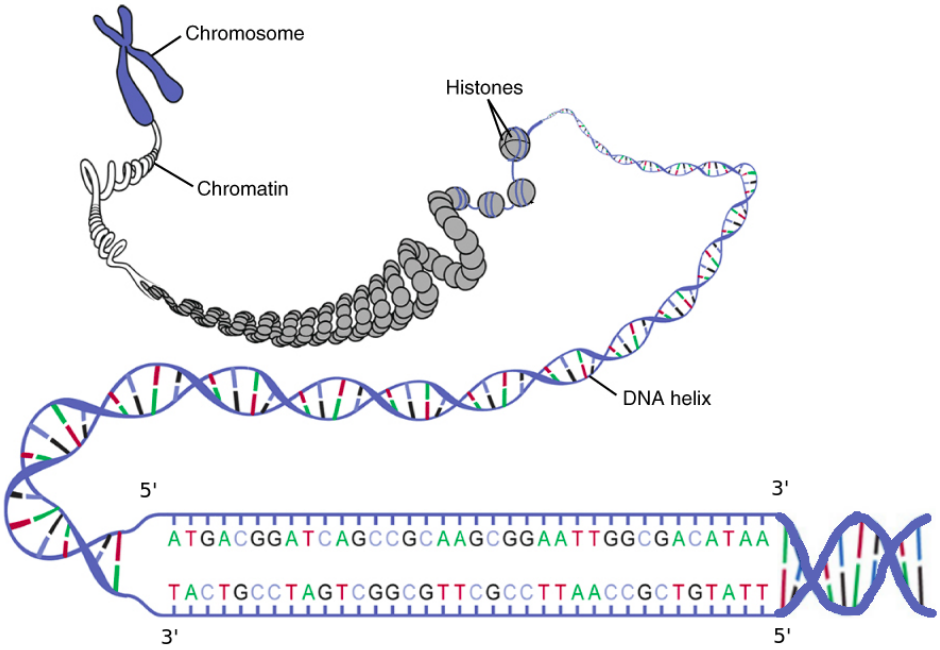


Figure 1.1: ***Structure of the DNA in a double helix.*** *A chromosome composed of double-stranded DNA coiled around histones. Each strand is a sequence of bases. The two strands are complementary, A corresponds to T and C corresponds to G.*

tory regions among the DNA sequence. This method was developed to detect genome-wide interactions that are mediated by some proteins of interest with a high resolution of one kilobase [FLP+09] while Hi-C is a genome-wide (probabilistic) assessment of proximity between all genomic regions within one megabase resolution [LAVBW+09].

Chromosomal compartments

One of the first discoveries using Hi-C technology in human is that all the regions in the genome can be classified into two different compartments noted A and B [LAVBW+09]. Regions that belong to a compartment will preferably interact with other regions of the same compartment. With this discovery, Lieberman-Aiden *& al.* [LAVBW+09] define compartment A as the set of active and open chromatin regions while B contains inactive and closed chromatin regions. Besides, they showed that these compartments are cell-type specific, in other words, A/B compartments vary between different cell types. A and B

compartments are about five megabases each and alternate along the chromosome [GD13]. Figure 1.1 shows regions of DNA loops that are clustered in two compartments: A (active in yellow) and B (repressed in violet)

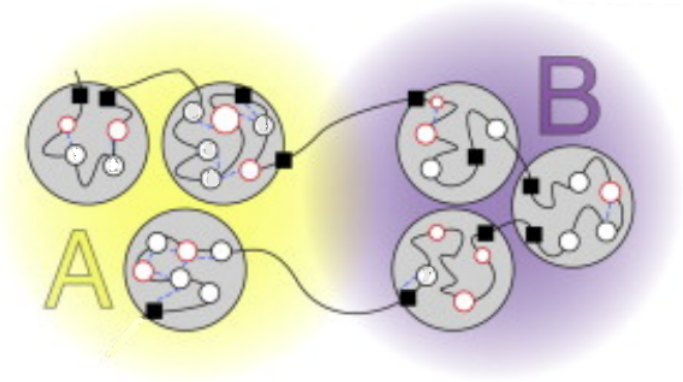


Figure 1.2: ***The A and B compartments****. Three-dimensional representation of the genome, where interactive regions are arranged in compartments (A or B). [Figure adapted from [GD13]]*

Topologically associated domains

On a lower scale than that of the compartments, Hi-C highlights the existence of another chromatin structure known as Topologically Associated Domains (TADs). TAD was defined by Dixon *& al.* in 2012 [DSY+12] as a genomic region in the range of megabases within which DNA physical interactions are persistent, as opposed to inter-TAD interactions, which are rare. The restriction of inter-TAD interactions is likely maintained by boundary activity between neighboring TADs [DGR16] (see Figure 1.3). Each chromosome is divided into multiple TADs, and the human genome is composed of almost 2000 TADs. Dixon *& al.* also showed that TADs are stable and conserved between cell types as well as between different species [DSY+12]. One TAD belongs to either active A or repressed B compartments [RIGP18]

TADs contain smaller subTADs that are not necessary conserved among cell types [PCSS+13] (see Figure 1.3). An example of this is presented in Figure 1.3, where X, Y, and Z are different regions equally distant (in term of bases), but X and Y are located in the same TAD while Z is in the neighbor TAD. The frequency of interactions between X and Y is

higher than that between Y and Z (Figure 1.3)

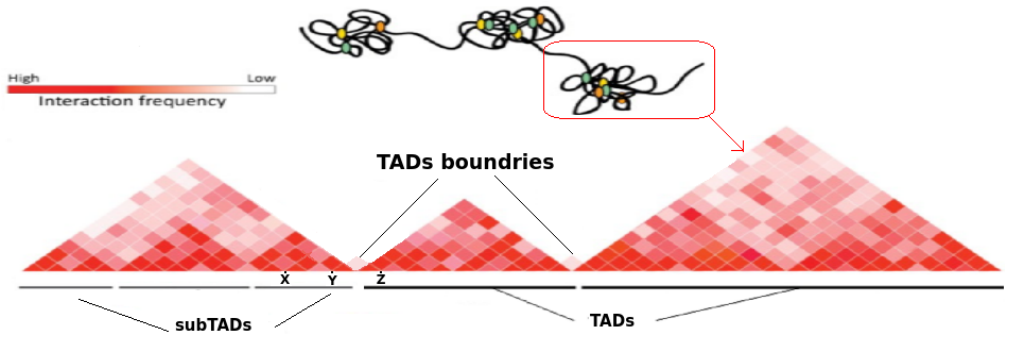


Figure 1.3: ***Visualization of a region of one chromosome.*** *TADs are the domains of DNA interactions and are represented here by the red triangles. SubTADs are also displayed. Red refers to high interactions while white refers to no interactions. X, Y, and Z are three equidistant regions. However, X and Y are in the same TAD while Z is in the neighboring TAD. X and Y show a high interaction frequency (red) while Y and Z show very low interaction frequency (white)*

Isochores

In 1981, Bernadi *& al.* [CSMB81] developed a strategy to understand the organization of the genome. This strategy relied on a fundamental property of the DNA: the base composition especially that GC content is not homogeneous along the DNA. They defined "isochores" as large regions of DNA (greater than 300 kb) that can be separated in two major classes: i) Heavy isochores with a high frequency of GC and ii) light isochores with a low frequency of GC. Years later, in 2017, Jabbari and Bernardi [JB17] showed that isochores are the genomic structure that underlies TADs.

1.1.1.2 RNA and the central dogma

Ribonucleic Acid (RNA) is a single-stranded copy of the DNA that can be transported from the nucleus to the cytoplasm. A specific sequence of DNA known as a *gene* (see Section 1.1.2) is transcribed to an RNA sequence using a complex of proteins including RNA polymerases. RNA is complementary to the DNA strand from which it is transcribed: the nucleotide composition of the RNA molecule is identical to that of the DNA except for the substitution of the thymidine (T) by uracil (U).

There are different types of RNA with various functions in the cell: i) Non-coding RNA (ncRNA,microRNA, ...) that can fine-tune gene expression. ii) RNAs implicated in the translation process (transfer and ribosomal RNA: tRNA & rRNA). iii) Messenger RNA (mRNA) that is translated to form specific proteins. mRNA is an intermediate molecule in the central dogma of life. The central dogma of molecule biology explains the flow of genetic information, from DNA to RNA, to make a functional product: the protein. It is often stated as "DNA makes mRNA and mRNA makes proteins". The central dogma is presented in Figure 1.4 where the coding gene in DNA is transcribed to messenger RNA that carries the information necessary to form a protein by a process known as translation.

Even though for a long time the central dogma was a fundamental principle that information flows from DNA to RNA to protein, novel discoveries are challenging this classical concept. In fact, hundreds of thousands of noncoding RNAs (ncRNAs) are now identified ([HRH+17], [HFG+12]). These RNAs were once written off as junk RNA. However, it is well known now that ncRNAs have critical regulatory roles in diverse molecular networks. With this revelation, the depiction of the genome has expanded to include thousands of non-coding genes that produce RNAs without producing proteins as their functional products ([HRH+17], [HFG+12],[Koo12], [MS15]).

1.1.2 Genes

The basic elements of hereditary information in DNA are genes. Genes were first defined by Johannsen in 1909 [Joh09] as the physical and functional unit of biological inheritance, but its meaning has been evolving since. Later, the concept of a gene was imposed as a sequence of DNA that conduct RNA synthesis that code for one protein. However, it is known that a gene is a sequence of DNA that can code for more than one protein. Besides, not all genes are protein-coding (in particular, miRNA, long non-coding RNA, ...). Thus a gene can be defined as a sequence of DNA that produces at least one RNA molecule with known, or unknown function [HFG+12].

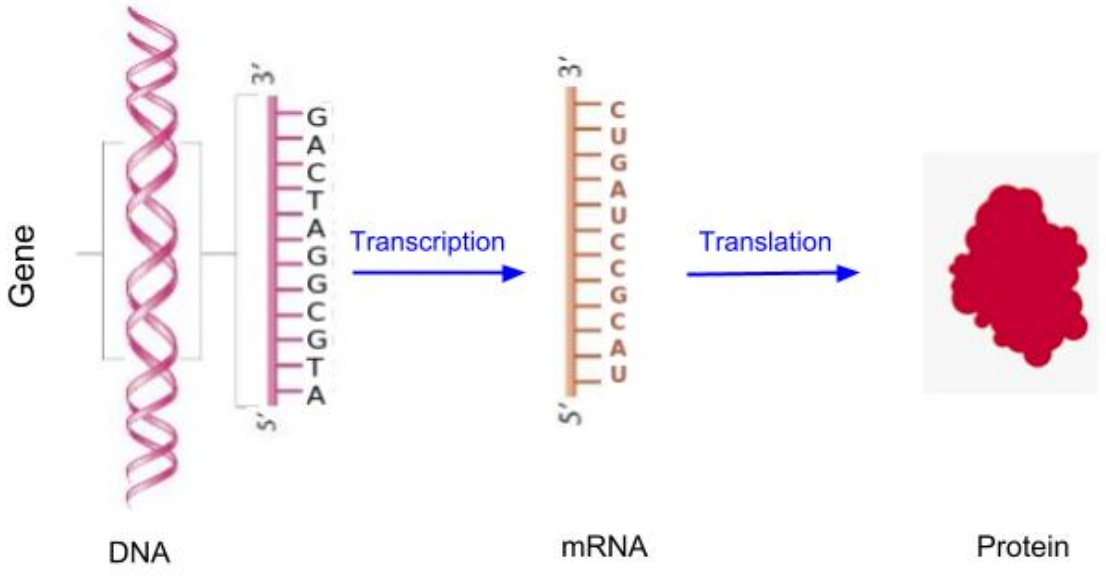


Figure 1.4: ***The central dogma.*** *From a double stranded DNA, genes will be transcribed to mRNA of single strand where A is transcribed to U, C to G and vise-versa. Then mRNA is translated to form a protein.*

1.1.2.1 Diversity of genes

The diversity of RNAs (Subsection 1.1.1.2) results from the variety of genes. Gene transcribed into RNAs not translated into proteins are known as non-coding genes. Protein-coding genes, on the other hand, are transcribed into Messenger RNAs. These genes represent around 42% of all genes annotated [HRH+17].

1.1.2.2 Protein coding genes

In this thesis, we will focus on protein-coding genes (referred to later as genes). The human genome contains around $20,000 - 25,000$ genes [HRH+17] with a median length of 12,426 bases. Each gene harbors one, or several transcription start sites (TSSs) and the most upstream is referred to as the gene start. The body of the gene is composed of two elements: exons and introns.

1. **Exons** are translated into proteins. These regions are retained in the mRNA after its maturation (splicing see introns). The number of exons, as well as their length and

composition, vary from a gene to another. The average number of exons in human genes is about $8-10$ and the average exon length is about 170 bases [SCK04]. A gene starts and ends with exons, these exons can intersect partially or totally with UTR regions (see Figure 1.5). Exons contain the coding DNA sequence (CDS). The rest of the exonic sequences, located upstream and downstream CDS, is referred to as 5'UTR and 3'UTR respectively (explained below).

2. **Introns** are sequences separating exons ([CGBR77], [BMS77]) (see Figure 1.5) and are present in immature pre-mRNA. These regions are transcribed then removed during the maturation of RNA in a process known as splicing [SL87]. Introns play a role in gene regulation ([CC12],[Ros08]). An average human gene contains about $6-9$ introns and the average intron length is about 5419 bases. A small proportion (5%) of introns is large (more than 200,000 bp) [SCK04]. The total length of introns cover almost 50% of the total length of the genome.

3. **UTRs** for Untranslated Transcribed Regions. Each gene has 2 UTRs: i) Upstream (on the 5'END) noted 5'UTR. In average, the length of this regions is around 150 bp [LM15] ii) Downstream (on the 3' END) noted 3'UTR with an average length of 350 bp [LM15]. These regions play a role in gene regulation ([RKC+13], [LML+13], [LQLM10]) as well as in mRNA stability [MP11].

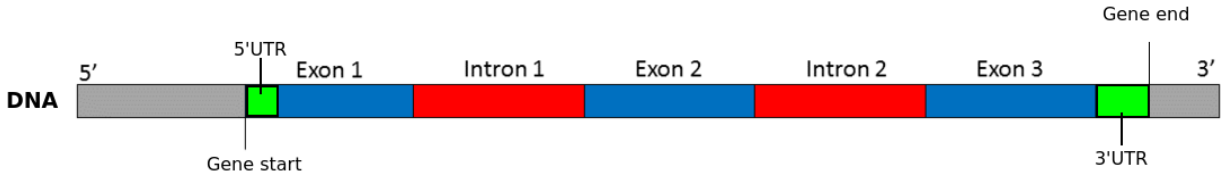


Figure 1.5: ***Gene structure.*** *A coding gene with a start (on 5') and an end (on 3'). Exons are presented in blue, introns in red and UTRs in green.*

1.1.3 Gene Regulation

Although all the cells of an organism contain almost the same DNA [com16], they still carry specialized biological functions. This is due to differential gene expression and regulation in each cell. Genes are not expressed in the same way in all cells [Dra03]. Some

genes known as *Housekeeping genes* are expressed all the times in all cells. However, the changes in the expression level of some particular genes in a specific cell, at a specific time, determine its characteristic and functions. This concept is the root of the complexity and the diversity of the organism. Gene expression varies from highly expressed to unexpressed at all. Quantifying the level of gene expression is the subject of different studies. Gene expression regulation occurs at different stages that include, transcription, translation, and degradation. The combination of all these processes determines when and where specific genes are activated, and the number of proteins or RNAs produced.
In this thesis, we are interested in the transcriptional and post-transcriptional regulation of gene expression. Figure 1.6 shows the detailed processes of transcription and post-transcription regulation of a gene. In Sections 1.1.3.1 & 1.1.3.2, we present the different regulatory elements illustrated in the Figure 1.6

1.1.3.1 Transcription regulation

Transcription is the first step of gene expression. During this process, the DNA sequence of a protein-coding gene is transcribed (copied) into a pre-mRNA. The transcription process is divided into three stages, i) initiation: beginning of the transcription (from the TSS) ii) elongation, which is the process of copying the gene into RNA and iii) termination: the end of the transcription (at gene end). Different elements control each of these stages, and in this section, we present the elements controlling mostly initiation and elongation. Termination is a poorly understood process.

Regulatory regions

An important element in transcription is regulatory regions. They are non-coding DNA sequences that are targets of RNA polymerase and other regulatory molecules involved in the process of gene transcription. These regions regulate the start and the level of transcription. Next, we present two types of regulatory regions: promoters and enhancers.

Promoters:
Promoters are a regulatory region located around the transcription start sites (TSSs) of the

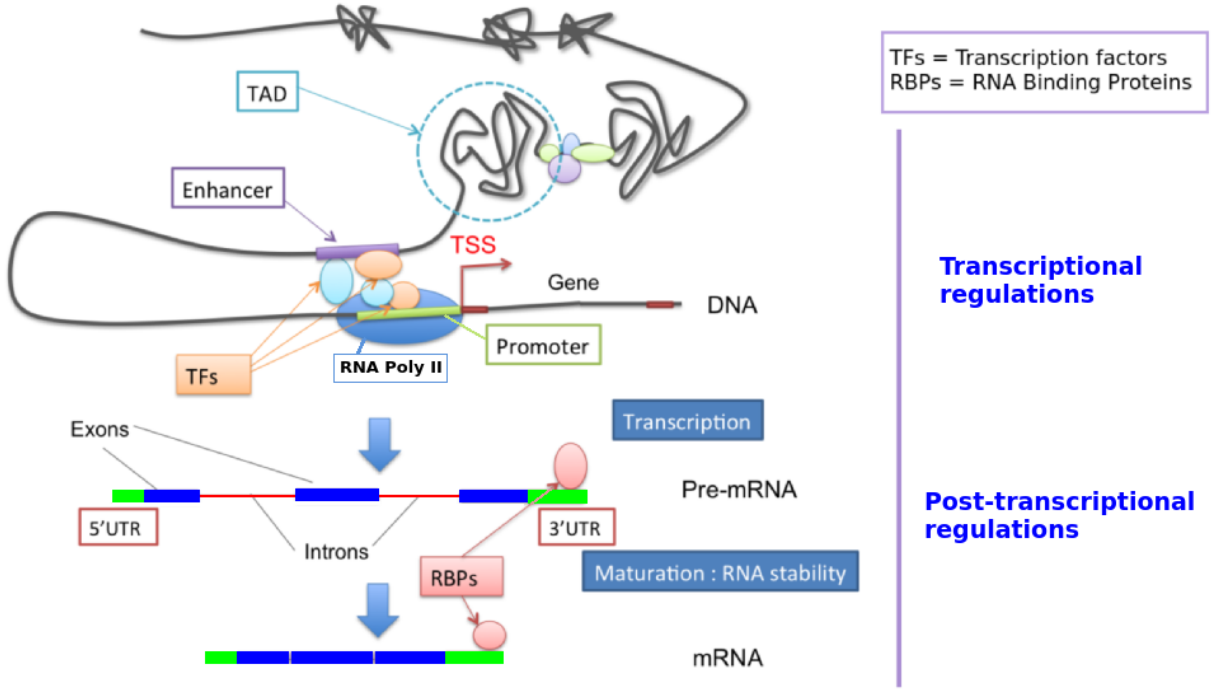


Figure 1.6: ***Different levels and mechanisms of gene expression regulation*** *From the DNA, a gene is first transcribed to pre-mRNA by RNA polymerases with the help of transcription factors (transcriptional level). At the post-transcriptional level, the pre-mRNA undergoes a process of maturation that will lead to a stable mRNA ready to be transported to the cytoplasm. All the mechanisms and biological elements are discussed below.*

gene (see Figure 1.6). Promoters contain specific DNA sequences which bind polymerase as well as other regulatory proteins called transcription factors (TFs) (see Section 1.1.3.1). Promoters harbor specific sequence features. One of these features is CpG ($Cytosine - phosphate - Guanine$) islands: short regions of DNA with an average length of 1,000 bp (500-4,000 bp), which are relatively GC-rich (65% GC content) compared to the whole genome (40% GC content). In fact, in humans, about 70% of gene promoters contain a CpG island [DB11]. Two other specific features are also important: The TATA box, which is a DNA consensus sequence of TATA repetition located 25 nucleotides upstream the TSS. This sequence is in part responsible for recognition of RNA polymerase II [Her93]. Finally, the initiator element (Inr) is a 17 bp sequence located at the TSS and has a similar function to that of the TATA box [XYF+07].

We define the position of a promoter in function of the TSS of a gene. However, a gene can have multiple TSSs [HFG+12] thus multiple promoters. The average number of TSS in the human genome is five [HFG+12].

Enhancers:

Enhancers are short nucleotide sequences in DNA that increase the rate of transcription of particular target genes. They act on promoters by binding regulatory elements and bringing them closer to promoters by a phenomenon known as DNA looping (see Figure 1.6). Although enhancers and promoters share many properties ([And15], [DS18]), enhancers can carry out their effects regardless of transcriptional orientation. Also, they can be both proximal (within the 5'UTR, the introns or the 3'UTR) and very distal from the target gene (Megabase order) [PBD+13].

Epigenetics

Epigenetics are the modifications at the genomic activity that occur without changes in the DNA sequence. Epigenetic is interested in a level of information that defines how genes will be expressed or not expressed in a cell.

Epigenetic modifications are natural and essential to many organism functions, but if they occur improperly, there can be major adverse health and behavioral effects [MA16]. Many types of epigenetic modifications have been identified including DNA methylation, histone modifications, and others. Next, we present some epigenetic modifications that can occur on regulatory regions, as well as their influence on gene expression and transcription.

DNA methylation

DNA methylation is a stable epigenetic modification that regulates gene expression. DNA methylation is the addition of a methyl group at cytosines (C) that occurs in the context of CpG di-nucleotides [SB08]. The human genome has a bimodal distribution of CpG methylation: i) most of the genome that is highly methylated ii) few regions with a CpG island enrichment that are not methylated. CpG islands mainly co-localize with promoters, the transcription regulatory unit of a gene.

It has been showed that DNA methylation of CpG islands functions to maintain a repressed

chromatin state and silence promoter activity [SB08]. The effect of the DNA methylation on gene transcription seems to depend on the CpG content of the promoter. Weber *& al.* [MWR+08] showed a clear anti-correlation between transcription of a gene and DNA methylation of a CpG-rich promoter. On the other hand, they showed that even though methylation of a CpG-poor promoter can repress transcription, sometimes the corresponding gene is actively transcribed.

Histone modifications

Histone modifications are post-translational modifications (PTMs) that take place on the tail of the histone and affect the structure of the chromatin. These modifications regulate gene expression by organizing the genome into active regions where DNA is accessible for transcription, or inactive regions, where DNA is more compact and less accessible for transcription. There are different types of histone modifications such as mono-, di-, and tri-methylation, acetylation, phosphorylation, and others. Each modification can lead to either the activation or inhibition of gene expression. Histone acetylation, for example, is often associated with transcriptional activation, while histone methylation can promote both transcriptional activation and inhibition. Each modification has a specific signature, for example, the trimethylation of the lysine 4 of histone 3 noted H3k4me3 is a particular modification of chromatin at the level of promoters that control actively transcribed genes in eukaryotes.

DNA accessibility

The binding of RNA polymerase and other regulatory elements is associated with DNA accessibility, in other words to the state (open or closed) of the chromatin near a particular gene. DNA accessibility is controlled by epigenetic regulations and can be measured, for example, by DNase I hypersensitivity [Wu80]. Deoxyribonuclease I (DNase I) is an enzyme that can cut free DNA. In figure 1.1 regions between two compacted histones are known as DNase I hypersensitive sites, short regions of chromatin that are highly sensitive to cleavage by DNase I [Wu80]. These sites are often considered as marks for transcriptionally active regions of the genome [WZW+12]. DNase-seq [KCLE81] is a method used to detect DNase I hypersensitive sites.

Transcription factors

Transcription factors (TFs) are molecules that control the activity of a gene by determining whether it is transcribed into RNA. Transcription factors control when, where, and how efficiently RNA polymerases function. TFs recognize specific short DNA sequence located on genes regulatory regions (promoter and enhancers). This short sequence of length between 5 and 30 nucleotides are known as transcription factors binding sites (TFBSs) or motifs. One TF can recognize several TFBSs that can differ in certain nucleotides. The representative model of motifs will be extended below.
TFs can be distinguished into two types: i) General transcription factors (GTF) needed in general for transcription to occur. These TFs form a complex called pre-initiation complex that binds specific sequences of the DNA especially the TATA box and the Inr. ii) Gene-specific TFs that are responsible for differentially regulate gene expression.
Specific transcription factors are often crucial in initiating gene expression that results in major developmental changes. A TF can be either an activator or an inhibitor of the transcription. An activator boosts gene transcription and helps the recruitment of RNA polymerase, while an inhibitor decreases transcription and/or impedes the binding of the activators and RNA polymerase. Although binding of TFs is a necessary factor in transcription, it is not in itself sufficient. Thus, following DNA binding, the factor must interact with other factors to form a complex and recruit RNA polymerase II (see Figure 1.6).

Motifs and PWMs A motif is an abstraction that models the sequence preferences of DNA binding proteins. The motif is built from multiple sequences known to be bound by the transcription factor (see Figure 1.7 a). Different models of representation have been elaborated. The simplest kind of motif representation is the consensus sequence [PMP04] *i.e.* a string of nucleotides that represents the most abundant nucleotides in each position of the protein's binding site. Variability between nucleotides at specific positions can be considered by using the IUPAC symbols [ZSRU91] to describe the motif (see Figure 1.7 b). However, this model is not flexible enough, since proteins often display variations in binding

specificities, besides, this is a deterministic model that presents a loss of information when compared with the collection of binding sites sequences[WS04]. Also, in positions where a base is not conserved (for example it can be T or A or C), the model does not give any information about the probability of having each of the three bases.
A visualization model was then introduced by Scheinder & al [SS90] to characterize each motif by its sequence logo (see Figure 1.7 e). In this model, bases are stacked on top of each other in increasing order of frequencies, and the importance of a nucleotide at each position is related to it size.
Another common representation relatively simple yet flexible is a matrix of positions in the binding site versus nucleotides [Sto00]. In the matrix, each row represents one residue (A, C, G or T), and each column represents a position in a set of aligned binding sites. There are several types of matrix representation: i) A position probability matrix (PPM) (Figure 1.7 c) that shows at each position, the probability of having a specific base based on the alignment of known sites (Figure 1.7 a). A probability of 1 means the base is conserved.. ii) A position weight matrix (PWM) (Figure 1.7 d) which can be deduced from the PPM. A PWM holds the log-ratio of the probability to observe the nucleotide given the motif model (PPM), and the probability to observe the nucleotide given the background model of the frequencies of each nucleotide in the genomic context.
Note that each motif has a specific length (number of bases) as well as a number of known sites to be aligned. Figure 1.7 illustrates an example of a motif with 14 bases length and eight known sites.

Direct interaction with the gene

RNA polymerase is an enzymatic complex responsible for the synthesis of RNA. In the human genome, three types of RNA polymerase exist I, II and III. Each type is specialized in the transcription of certain classes of genes. The RNA polymerase binds specific binding sites on the promoter and forces the DNA to unwind to expose the two strands of DNA. The RNA polymerase uses a single-stranded DNA template to synthesize a complementary strand of pre-mRNA, it builds a pre-mRNA strand in the 5' to 3' direction. The transcription of protein-coding genes into mRNA is produced using RNA polymerase II.

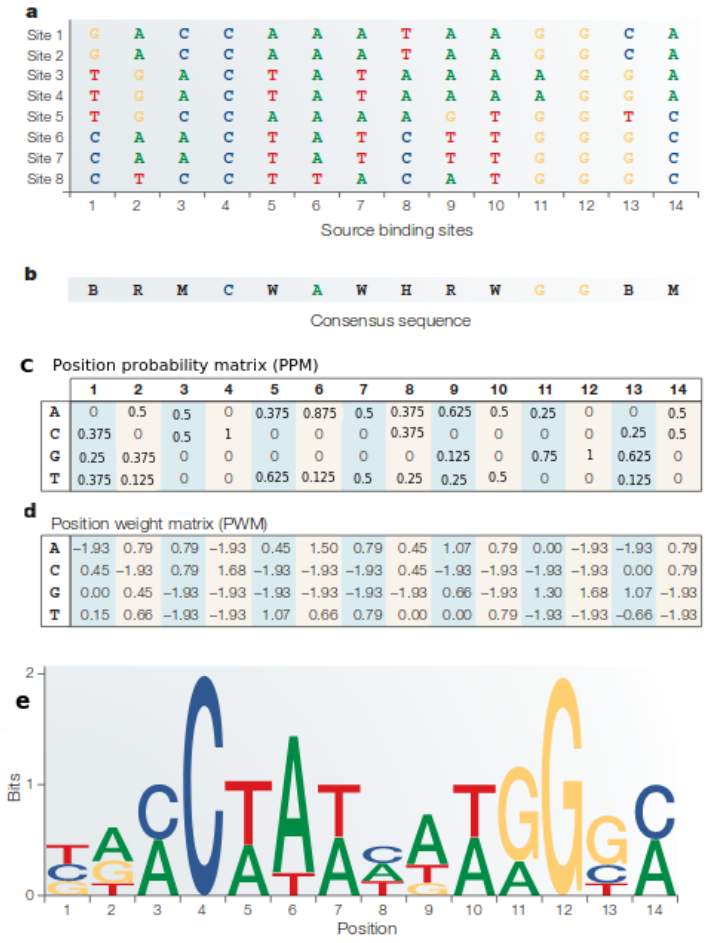


Figure 1.7: ***Different Models of representation of a motif with 14 bases.*** *a) Eight different binding sites of a TF collected from the literature and aligned. b) A consensus sequence model using the IUPAC symbols. c) Position Probability matrix (PPM) with 14 column one for each position and four rows one for each base. The probability of having C on the first position is 0.375 while there 0% chance to have an A. d) Position weight matrix (PPM) calculated using the PPM. e) Sequence logo model where the size of the base at a position represents its frequency. Figure adapted from [WS04]*

The RNA pol II propagates along the copied DNA strand to form the pre-mRNA similar to the coding strand with U instead of T (elongation). The RNA pol II continues transcribing the DNA until it gets to the gene end (termination). In the human genome, the binding of the RNA poly II is controlled by transcription factors and epigenetic modifications [LJZ15].

1.1.3.2 Post-transcription regulation

At the RNA level, post-transcriptional controls of gene expression and RNA maturation are exerted through splicing, RNA-binding proteins regulation, and mRNA nuclear export. One of the first steps of RNA maturation is the splicing process in which introns are removed from pre-mRNA and exons are joined together to form the mRNA that will be transcribed into proteins. Brenner *& al.* [LBS+07] showed that pre-mRNAs in humans undergo alternative splicing in 95% of genes. The process of alternative splicing can gen-

erate multiple transcripts from the same gene, thus potentially increasing the proteomic diversity and then influencing gene regulation. In addition to this, the importance of alternative splicing is further supported by its link with development and disease [OC07].
On the other hand, RNA-binding proteins (RBPs) can co- or post-transcriptionally regulate the fate of RNAs. The vast majority of RBPs appear to bind target sequences in single-stranded RNA (see Figure 1.6) ([RKC+13], [LQLM10], [AOA06]). Some RBPs can also recognize specific sequences in their RNA targets through small non-coding RNAs *i.e.* microRNAs (miRNAs), and this recognition eventually leads to RNA degradation [SCFK14].
Finally, the regulation of gene expression relies partially on the controlled exchange of molecules between the nucleus and the cytoplasm. The RNAs produced in the nucleus have to be exported, either to be translated into protein or to mature into functional particles, whereas factors implicated in transcription regulation have to be imported into the nucleus. Export of mRNAs is unique as it is extensively coupled to splicing [RDS04].
The RNA regions involved in post-transcriptional regulations mostly correspond to the 5' and 3' untranslated regions (5'UTR and 3'UTR) ([RKC+13], [SCFK14], [GHT14]). The coding DNA sequence (CDS) also plays a key role in post-transcriptional regulation [RH05].
Intronic sequences have also been reported to affect gene regulation in many ways, from transcription to RNA stability ([CC12], [Ros08]).

1.1.4 Gene deregulation in cancer

1.1.4.1 Introduction to cancer

Cancer is a term that refers to a collection of diseases that have the common feature of uncontrolled cell growth and invasion. The general term "cancer" does not refer to a disease but applies to a range of pathologies that can affect any part of our body. There are more than 100 different types of cancer usually named according to the cell type.

In 2016, The International Agency for Research on Cancer (IARC) declared 14 million new incidences worldwide, of which almost 8 million deaths occurred in one year [O+05]. According to the "Institut National du cancer" (INCa), cancer is the principal cause of

death in France. They estimated almost 400,000 new cancer cases with 150,000 deaths in 2017.

Cancer starts when a cell acquires a series of modifications that collectively change a normal cell into a cancerous cell, which divides uncontrollably and may eventually spread throughout the body. The first indications of the significant role of genetics in cancer development date from the beginnings of the XX^{th} century. Nowadays, it is demonstrated that cancers are mostly genetic pathologies [Wei07]. Cancer-causing modifications can occur on different levels: epigenetic modifications and genetic mutations on the DNA sequence. These modifications may occur on all the genome, and they can lead to the development of cancer only when occurring on specific genes implicated in cancer called cancer-genes.

1.1.4.2 Cancer genes

Cancer genes can be classified into two types, proto-oncogenes/oncogenes and tumor suppressor genes. In the normal state of the cell, a balance exists between the expression of these two types and their modifications/mutations may break this balance and favors carcinogenesis.

Proto-oncogenes and oncogenes:
Proto-oncogenes are a group of genes that cause normal cells to become cancerous when they are mutated. Proto-oncogenes code for proteins implicated in growth factors, transcription factors, and regulators of cell death. All of these processes are important for normal human development and the maintenance of tissues and organs.
Oncogenes induce positive regulation of proliferation and differentiation. Oncogenes typically increase cell division, decrease cell differentiation, and inhibit cell death. These phenotypes define cancer cells.

Tumor suppressor genes:
Tumor suppressor genes are involved in the negative regulation of proliferation and act either by inhibiting signaling pathways that may favor oncogenesis or by activating path-

ways that can block oncogenesis. We regularly observe inactivation of tumor suppressor genes in cancer cells [dL94].

1.1.4.3 Epigenetic modifications

The first human disease to be linked to epigenetic modifications was cancer. Feinberg & Vogelstein [FV83] (1983) found that diseased cells from patients with colorectal cancer had less DNA methylation than the normal cells of the same patients. Modified histones complex disrupt the pattern and levels of histone marks and consequently deregulate the control of chromatin-based processes (activation or inhibition of cancer-genes transcription), ultimately leading to the development of cancer [SH10]. In the same concept, DNA methylation can also be a cause of cancer. Effectively, specific hypermethylation of CpG islands silences tumor suppressor genes while a global hypomethylation of the genome favors oncogenes activation [Est08].

1.1.4.4 Mutations

When talking about mutation, one should first introduce genetic variations. Genetic variations are changes to our DNA that can modify genetic information. These changes are always happening, and they can be caused by exogenous exposures such as chemicals and radiations, as well as by endogenous mechanism during cell division. The most common way through which simple variations occur is DNA copying errors. Given the number of nucleotides in the DNA to be copied at each cell division, mistakes are inevitable. The rate of variation in eukaryotes is 1 per every 100,000 nucleotides [Nog18]. Even though most of the mistakes are repaired or eliminated by DNA processes, few mistakes do get through each and every cycle [Pra08]. Every cell has thousands of genetic variations that make it a little bit different from its mother cell [JWPP00].

Types of variations:

The genetic variation occurs at vastly different scales, from a single base in a genome to entire chromosome structure. Mutations can be derived from single nucleotide variations

and small insertions and deletions (SNVs) or from copy number variations.

Single nucleotide variations can be: i) punctual variations, *i.e.* substitution of one nucleotide by another, which constitute 95% of genetic variations. ii) Indels variations *i.e.* variation by deletion or insertion of a small sequence in the genome. These variations may occur in different regions of the DNA: genes, promoters, intronic regions, and others.
The other type of mutations is copy number variation (CNV). Genes are presents in two copies in a genome, one from each parent. For some genes, however, the copy number varies from this norm, and they can be present in one (deletion), three, or more than three copies (amplification). This is known as copy number variation of DNA segments, and it is ranging from kilobases (kb) to megabases (Mb) in size [SLT+04].
When these variations occur in more than 1% of the population with no negative effect, they are known as Polymorphisms. In particular, the substitution of one nucleotide, in this case, is known as Single Nucleotide Polymorphisms (SNP) [Sha13].

However, when the genetic variation rate is often identified in the same disease (for example cancer), these variations are known as mutations [KPEF15]. Mutations can be classified into two classes: i) passenger mutations, *i.e.* do not cause diseases especially cancer. Otherwise, when one or more of these mutations occur in a cancer gene (proto-oncogene or tumor suppressor gene), they become driver mutations and lead to cancer development [PM15].

Even though it is known that somatic mutations can be generated by endogenous (DNA machinery) or exogenous (tobacco, ultraviolet, ...) factors, the understanding of the mutational processes that cause somatic mutations in cancer is limited. A number of "signature" characterizes each type of cancer. A signature is a combination of different somatic mutations generated by different mutational processes. In 2013, two back to back studies by Lawrence *& al.* [LSP+13] and Alexandrov *& al.* [ANZW+13] proposed methods to identify cancer-associated signatures and evaluate the heterogeneity of the mutational process among cancer types.
Alexandrov *& al.* [ANZW+13] were able to distinguish different elements (age, transcrip-

tion, type of mutation) that are correlated to the cancer signatures. On the other hand, Lawrence *& al.* proposed a method using the background mutation rate (*i.e.* the per base rate at which new mutations occur) to identify cancer genes with a low false positive rate.

1.2 Statistical methods

For the methods presented below, let n and p be respectively the number of individuals and that of variables. The notation of variables used in this section is: i)Y is an $[n \times 1]$ vector of the response variable and ii) $X = (X_1, \ldots, X_p)$ an $[n \times p]$ variable matrix where $X_{j\,(j=1,\ldots,p)} = (x_{1j}, \ldots, x_{np})$ is a column vector corresponding to the observation of variable j for the n individuals.

1.2.1 Parametric regression

1.2.1.1 Linear regression

A linear regression is a statistical method that aims to find a linear relationship between a response variable Y and one or more predictive variables. The terminology linear regression was introduced by Francis Galton in 1886 [Gal86]. Although the method used to estimate this type of model in general, known as Least squares, was introduced long before by Legendre in 1805 [Leg05].

A linear regression can be written in a matrix form as in Equation (1.1)

$$Y = X\beta + \epsilon \tag{1.1}$$

where X is the $[n \times p]$ predictive variables matrix, $\epsilon = (\epsilon_1, \ldots, \epsilon_n)$ is the vector of residuals. The vector of parameters $\beta = (\beta_0, \beta_1, \ldots, \beta_p)$ is the regression coefficients to be estimated. β_0 is the intercept or bias of the machine learning and is estimated to the mean of the response variable when predictive variables are standardized. In Equation (1.1), X is the matrix of variables with adding a vector 1 (corresponding to β_0). Matrix X is then written as :

$$X_{[n\times(p+1)]} = \begin{bmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1p} \\ 1 & x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & x_{n2} & \cdots & x_{np} \end{bmatrix} \tag{1.2}$$

The application of the linear regression model requires the validity of the three assumptions presented below:

1. For all $i = 1, \ldots, n$, ϵ_i are independent identically distributed $\epsilon_i \sim \mathcal{N}(O, \sigma^2)$
2. Homoscedasticity: Same variance σ^2 for all $\{\epsilon_i\}_{i=1,\ldots,p}$
3. No or little multicollinearity between $\{X_j\}_{j=1,\ldots,p}$
4. $n \gg p$

Each of these assumptions has its importance in the model estimation. First, from the first assumption, the normality of residuals is necessary for the tests of the nullity of coefficients and calculation of their confidence intervals. Zero means an ad hoc assumption used to simplify the calculation. On the other hand, the independence of the residuals leads to non-biased estimators (not valid otherwise) [SW15].
The importance of the second assumptions is related to coefficient estimation. Homoscedasticity establishes independence between residuals and the predictive variables. This is necessary for the least square that by definition accord equal weights to the variables [AS95]. Finally, high multicollinearity between variables affects the persistent of the estimations as well as the stability of the model. Besides, it can lead to overfitting [MPJ91]. Regarding the last assumption, its importance is explained later after the presenting the optimizer.

For estimation, let $\hat{\beta}$ be the vector of estimated coefficients and $\hat{Y} = X\hat{\beta}$ the predicted vector. The estimator $\hat{\beta}$ is obtained by the Ordinary Least Square (OLS) estimator that minimizes the residuals sum of squares in Equation (1.3) :

$$RSS = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{1.3}$$

This is a quadratic function and its minimum always exists but may not be unique. The explicit formula of $\hat{\beta}$ is presented in Equation (1.4):

$$\widehat{\beta} = \underset{\beta}{\operatorname{argmin}} \left(\sum_{i=0}^{n} (y_i - \sum_{j=0}^{p} \beta_j x_{ij})^2 \right) \tag{1.4}$$

when $X^T X$ is invertible, the unique solution $\hat{\beta}$ is given by Equation (1.5).

$$\hat{\beta} = (X^T X)^{-1} X^T Y \tag{1.5}$$

This estimator is convergent and known as the Best Linear Unbiased Estimator [Ait36].

The importance of the last assumption presented previously ($n \gg p$) intervene in the computation of $\hat{\beta}$. If p is higher than n, there is no longer a unique least squares coefficient estimate. The inverse of $X^T X$ does not exist, and the method is not useful [Clo09].

When fitting a linear model, the aim is not only to calculate prediction but also to select a subset of variables that are significant to explain the response variable. In literature, different methods exist for variable selections for linear models [Hoc76]. These methods showed high performance in low dimension studies. When the study is in high dimension, especially when p is high, the methods in [Hoc76] were not very efficient. For this reason, the ultimate model (see Section 1.2.1.1) was proposed with the aim of variable selection especially when the number of variables is high.

1.2.1.2 Penalized linear regression

A variable selection model was developed by Robert Tibshirani in 1996 known as Lasso regression for Least Absolute Shrinkage and Selection Operator [Tib96]. In addition to its capacity of variable selection. This model was built to overcome the Curse of dimensionality, in other words, the model is efficient in high dimensional problems when the number of variables is higher than the number of individuals ($n \ll p$) [Tib96]. Lasso optimization is an extension of the least square optimizer. The solution of lasso is obtained by minimizing the residual sum of squares (From Equation (1.1) with a norm ℓ_1 penalty term weighted by the regularization parameter λ as in Equation (1.6).

$$\hat{\beta}_{LASSO} = \underset{\beta}{\operatorname{argmin}} \left(\sum_{i=0}^{n} (y_i - \sum_{j=1}^{p} \beta_0 x_{ij})^2 + \lambda \sum_{j=0}^{p} |\beta_j| \right) \tag{1.6}$$

The solution can also be seen as a minimization of the RSS under a constraint of type norm ℓ_1. The solution is then :

$$\hat{\beta}_{LASSO} = \underset{\beta}{\operatorname{argmin}} \left(\sum_{i=0}^{n} (Y - \sum_{j=0}^{p} \beta_j x_{ij})^2 \right) \quad \text{under the constraint} \sum_{j=0}^{p} |\beta_j| \leq t \tag{1.7}$$

where t is a parameter to control the level of regularization. This constraint penalizes the regression coefficients with a norm ℓ_1. Parameters λ and t are related by an explicit relation that depends on the training data. The solution with a penalty allows a moderate increase in the residuals sum of squares but with a decrease of the standard error of each estimation. The choice of the ℓ_1 norm comes from the fact that it leads to variable selection by forcing the coefficients of the predictive variables with no effect to be zero [FB17]. Variable selection is an advantage of Lasso contrary to other norms, for example, ℓ_2 norm knows as ridge regression [HK70]. Figure 1.8 shows a graphical representation of the solution of a norm ℓ_1 versus norm ℓ_2 penalty in $\mathbb{R}^2$. The shape of the ℓ_1 penalty (square) leads to intersections with the axes and consequently to null coefficients. This is not the case for the ℓ_2 norm (Figure 1.8 right graph) where a circular shape leads to minimum variance which stabilizes the estimations [HTF01]. The Ridge optimization is useful when it comes to model performances, but it can not affect a variable selection (as for Lasso).

The choice of the value of λ is crucial and depends on the data. $\lambda = 0$ leads to the OLS solution and if $\lambda \longrightarrow \infty$ all the coefficients are set to zero. To estimate the model, a set of values of λ is defined and the value that minimizes the mean square error by cross-validation [RTL16] is chosen. The number of selected variables is inversely proportional to the value of λ. The higher the value of λ is, the fewer variables are selected [FB17]. An example of a Lasso penalized regression fitted on 20 predictive variables to predict a response variable Y is illustrated in Figure 1.9. The first graph Figure 1.9 (a) shows that when λ is higher (log scale), the number of selected variables is lower. For $\log(\lambda) = -5$ all variables are selected. On the other hand, when $\log(\lambda) = 0$ no variable is selected.

Universally, two values of λ are used: the one that minimizes the mean square error (λ_{min}), and the one that gives an error which is one standard error away from the minimum error [FHT10]. (λ_{1se}). λ_{min} is lowest than λ_{1se} which induces a model with a higher number of selected variables. Figure 1.9 (b) shows, for the same example as Figure 1.9 (a), the output of the cross-validation applied to select to best value of λ. λ_{min} and λ_{1se} are also presented in the graph. The model fitted using λ_{min} as the penalty parameter has ten non-zero coefficients and the one using λ_{1se} has only eight non-zero coefficients.

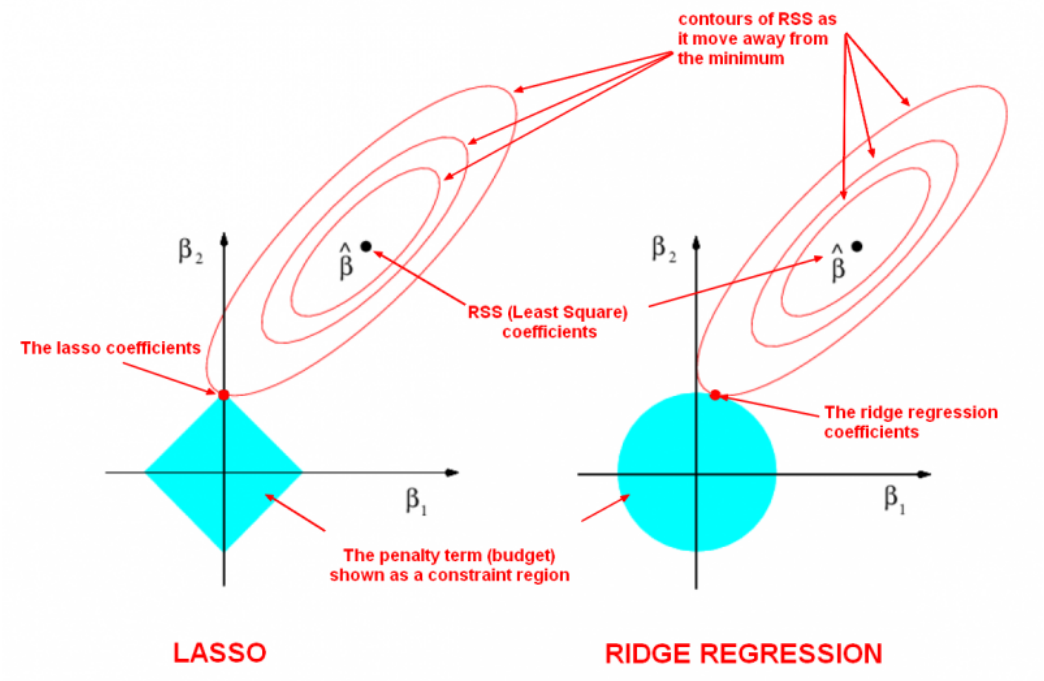


Figure 1.8: ***Estimation for lasso and ridge regression in*** $\mathbb{R}^2$. *Representation corresponds to a model with two variables with coefficients respectively* β_1 *and* β_2. *The ellipses represent the RSS as it increase from its minimum value* $\hat{\beta}$. *The square (lasso) and the circle (Ridge) represents the shape in* $\mathbb{R}^2$ *of the constraint of norm* ℓ_1 *and* ℓ_2. *The solution of each optimization is the intercept between the ellipse and the constraint. Figure inspired from The elements of statistical learning page 71 [HTF01].*

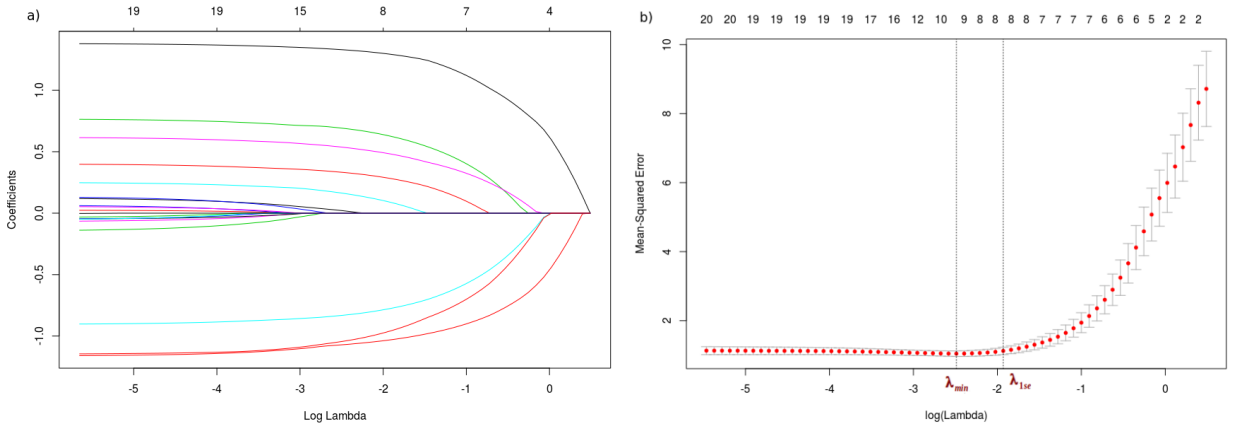


Figure 1.9: ***Representations of the output of a Lasso penalized regression model*** *fitted on 20 predictive variables. a) Path of the coefficients against the value of log(λ). Each curve represents one of the 20 variables. Values on the upper axis represent the number of nonzero coefficients. b) Cross-validation errors and their upper and lower confidence interval for each value of λ. The upper axis represents the number of selected variables. Vertical dotted lines refer to the values of λ_{min} and λ_{1se} respectively.*

1.2.1.3 Stability selection

This method comes in a logic continuity of the Lasso regression. It is used to select stable variables. In other terms, variables that are selected with high probabilities even after a slight changing in the training dataset. This method, introduced by Meinshausen and Bühlmann in 2010, [MB10] is based on a sub-sampling and a selecting algorithm.

The concept of the method is to find, among p predictive variables, the set $\{\hat{S}^{\text{stable}}\}$ of stable variables when randomly resampling the training data. A variable is considered as stable if its probability to be selected is higher than a defined threshold π_{thr} over a large number of resampled training datasets [MB10]. As for the Lasso penalized linear regression, one should identify a set regularization parameter Λ. The choice of Λ and its effect on the algorithm will be discussed later in this section. The stability selection is explained in Algorithm (1)

In their paper, Meinshausen and Bühlmann [MB10] show that when π_{thr} varies in a range of 0.7 and 0.9, the results are slightly affected. Also, they show that the choice of Λ do not

affect the set of selected stable variables dramatically, as long as Λ varies within reason. In fact, the set Λ can be chosen in a way that guarantees a bound on the expected number of false selections. In fact, some of the variables selected as stable may be false positives. With the aim of controlling the error of the method, let V be the number falsely selected variables and q_Λ the average number of selected variables. The per-family error rate is defined as the expected number of false positives $E(V)$ [MB10]. The error of the method is controlled either by π_{thr} or q_Λ as defined in Equation (1.8):

$$E(V) \leq \frac{1}{2\pi_{thr} - 1} \frac{q_\Lambda^2}{p} \tag{1.8}$$

Algorithm 1: Stability selection

Input : Response variable $Y_{[n\times 1]}$ and predictive variables matrix $X_{[n\times p]}$

Output: Set of stable selected variables and a matrix of selection frequencies.

Initialize : The Number of repeats B. The set of regularization parameter Λ. The threshold of selection π_{thr}

for $\lambda \in \Lambda$ **do**

  for $i = 1$ **to** B **do**

    - Draw from population, without replacement, a subset Z_i of $\frac{n}{2}$ individuals;
    - Attribute a random uniform weight for each variable;
    - Run a Lasso penalized regression on Z_i using the penalty λ;
    - Note $\hat{S}_i^\lambda$ The set of selected variables;

  end

  for $k = 1$ **to** p **do**

    Calculate a selection frequency ;

$$\prod_k^\lambda = \frac{1}{B} \sum_{i=1}^{B} \mathbf{I}_{\{k \in \hat{S}_i^\lambda\}} \ ;$$

  end

end

Construct the set of stable variables according to the following definition:

$$\hat{S}^{\text{stable}} = \{k : \underset{\lambda \in \Lambda}{max} \prod_k^\lambda \geq \pi_{thr}\};$$

return $\{\hat{S}^{\text{stable}}, \{\prod_k^\lambda\}_{\{\lambda\in\Lambda, k=1,\ldots,p\}}\}$;

1.2.2 Non-parametric regression

1.2.2.1 Regression Trees

Decision trees are non-parametric and non-linear models used to predict a response variable Y using a set of predictive variables X. The algorithm CART for classification and regression trees presented by Breiman in 1984 [BFOS84] is the most popular. The CART is a recursive binary partitioning of the variables X to predict the class or the continuous value of Y for classification and regression respectively.
Next, we will detail the different steps of the procedure to obtain a regression tree. In this thesis, we are only interested in regression questions (predictions of continuous gene expression). We will not detail classification.

Construction of the tree for regression:
The result is presented as a hierarchical tree. The start is with a root node including all individuals. At each partitioning, the method consider all the possible divisions of form $X_{j\,(j=1,\ldots,p)} \leq t$ where $t \in X_j = (x_{1j}, \ldots, x_{nj})$ and divide the node into two nodes (left and right). In each node, Y is fitted by the mean of the observations in this region according to the division variable. The best division (variable and threshold) is selected to minimize the cost function defined in Equation (1.9) [RRCB10]:

$$Cost = SS_T - (SS_L + SS_R) \tag{1.9}$$

SS_T is the sum of squares before the division, SS_L and SS_R are respectively the sum of squares in the left and right node.
The selected division is the one that induces the higher loss of error *i.e.* the one that minimizes the cost function. This criterion is known as *anova* because it reduces the inter-groups variance at each division [**?**].

All individual satisfying the selected division $X_j \leq t$ will be set on the left child node and the rest on the right child node. This procedure will be repeated for each child node until

having homogeneous leaf nodes or until attaining a stopping criterion.

Figure 1.10 illustrates an example for CART in $\mathbb{R}^2$ fitted on 40 individuals. For the root node, the algorithm select $X_1 < t_1$ as the best division creating a left child node A with 12 individuals almost homogeneous (small error) (Figure 1.10 a). For the rest of the data (in the right node) the algorithm selects the division $X_2 < t_2$ to create another nodes B with 8 individuals (Figure 1.10 b). The algorithm continues with the last division (Figure 1.10 c) to obtain two additional partitions C & D that are not as homogeneous as A & B. The algorithm stops and returns the decision tree in Figure 1.10 d.

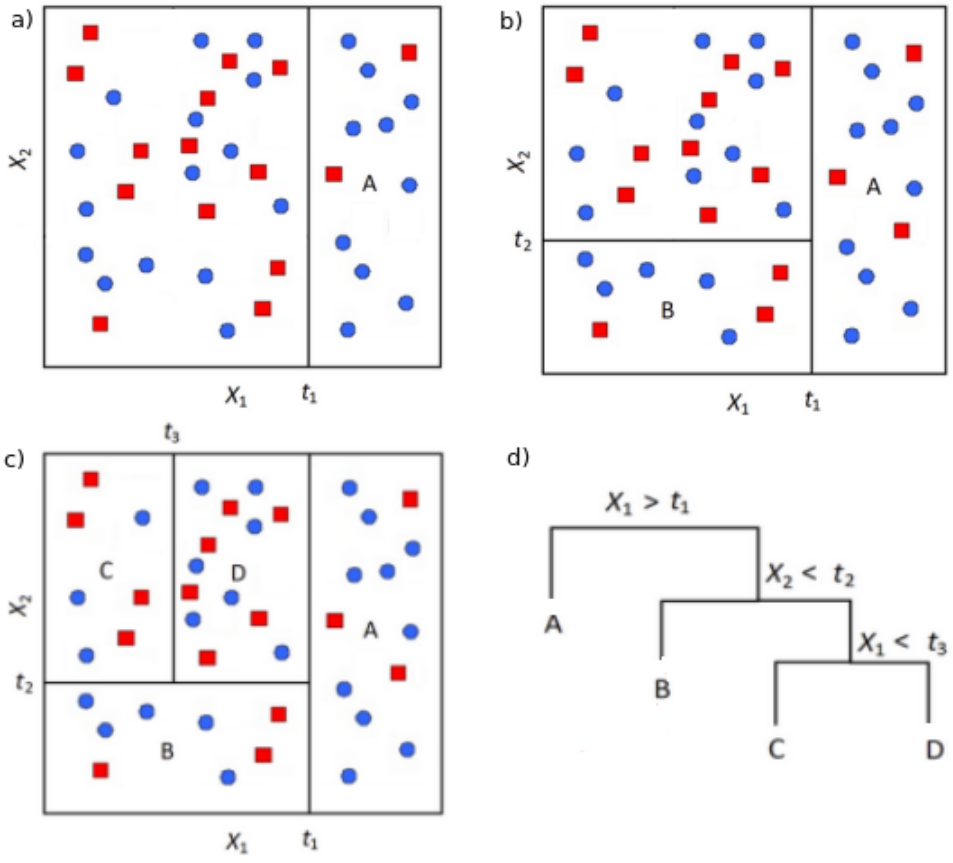


Figure 1.10: ***CART algorithm example in*** $\mathbb{R}^2$ *The data on this example are simulated. a) First split under the condition* $X_1 < t_1$ *b) The rest of the data are divided according to* $X_2 < t_2$*. c) Final division with* $X_1 < t_3$ *d) The results of CART algorithm in form of an hierarchical tree.*

Stopping criteria:

The more we progress in the partitioning (*i.e.* the deeper the tree is), the higher is the risk of overfitting. This algorithm needs to know when to stop. Different stopping criteria exist [Kra96], and one of the most common criteria is to define a minimum count of individuals for leaf nodes. If at an internal node, the number of individuals is less then the minimum the split is not accepted, and the node is considered as a leaf node. In general, the minimum is set to be at least the third of the number of individuals of the divided node.

Pruning the tree:

Another method proposed by Breiman to avoid over-fitting of the CART algorithm is the pruning of the tree [HTF01]. This method will determine the depth of the tree *i.e.* the number of internal nodes by using a complexity parameter. The pruning consists of three steps i) Grow a maximal depth tree T_0 where at each leaf node we have the minimum node size (in general 5). ii) Using the maximal tree T_0, define $T_1, \ldots, T_M$ all the sub-trees that can be pruned by the complexity parameter from T_0 (explained later). iii) Select the sub-tree that has the lowest mean square error on a test sample using cross-validation.

Define the sub-trees:

Let T_0 be the full tree and T any pruned sub-tree from T_0. T is obtained by collapsing a number of internal nodes [**?**]. First, one should calculate $R(T)$ the error committed with the sub-tree T. let $\tilde{T}$ the set of leaf nodes of T, $R(T) = \sum_{t \in \tilde{T}} (y_i - \bar{y}_t)^2$ where $\bar{y}_t$ is the mean of the observations in the leaf node t. The cost complexity function $C_\alpha(T)$ is defined by:

$$C_\alpha(T) = R(T) + \alpha|T| \tag{1.10}$$

where α is a tuning parameter that penalizes $|T|$ the number of leaf nodes in T. A higher value of α induces a smaller tree (less deep). The idea is to find for each α, the unique sub-tree $T_\alpha \subset$ T that minimize $C_\alpha(T)$. The value of α is estimated by cross-validation on k-folds.

1.2.2.2 Random Forest

Random Forest was developed in 1991 by Leo Breiman [Bre01] as an upgrade of the initial method of decision trees. Sensibility and instability are two big inconvenient for decision trees. Random Forest, constructed on the base of multiple decision trees overcome these limits and provide a new non-parametric powerful framework valid for regression as well as for classification.

Random Forests is also based on a bagging method [Bre96]. Bagging is an abbreviation of bootstrap aggregation is an ensemble method [Zho12] used to reduce the variance for those algorithms that have high variation. Bagging is usually very useful for CART.
The algorithm of construction of a Forest in the context of regression is presented in Algorithm (2).

Algorithm 2: Random Forest

Input : Response variable $Y_{[n\times 1]}$ and predictive variables matrix $X_{[n\times p]}$

Output: One Random forest

Initialize : *ntree*: number of decision trees used to construct the forest;

for $t = 1$ **to** *ntree* **do**

- *nobs: number of observations to train each tree;*
- *m: number of variables used at each node of the tree*;

Grow a full tree using *nobs* individuals and for each node choose the best cut when using only m predictive variables;

$\widehat{P_{tree}(t)} = (P_{tree}(1,t), \ldots, P_{tree}(n,t))$ is the vector of mean predictions of the individuals calculated on tree t.

end

Aggregate all the trees predictions to create the prediction of the random forest;

$\widehat{P_{forest}} = \frac{1}{\text{ntree}} \sum_{t=1}^{ntree} P_{tree}(t)$;

return the predictions $\widehat{P_{forest}}$.

Figure 1.11 shows an example of the algorithm 2 with $ntree = 3$ trees. Three trees are grown on three different subsets of *nobs* individuals and m variables each (bagging part). The resulted prediction in one leaf is an aggregation of the three predictions calculated on the different trees.

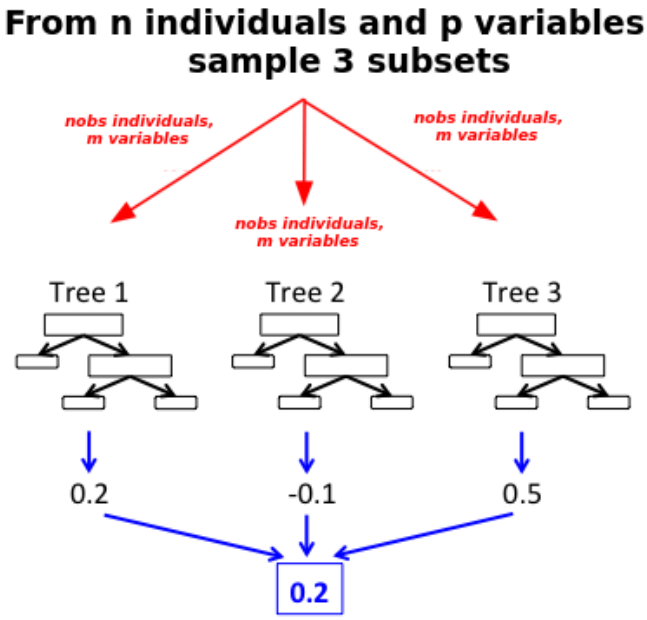


Figure 1.11: ***Example of the Random Forest algorithm.*** *For this example ntree is equal to 3. From the original subset with n individuals and p variables, three independent subsets with nobs individuals and m variables are selected. Tree 1, Tree 2 and Tree 3 are grown on each subset. The predictions of a leaf node of each tree is respectively* $0.2, -0.1, 0.5$. *The prediction from the random forest is the mean of three predictions i.e.* 0.2

1.2.2.3 Local regression: Loess

Loess for LOcal regrESSion is a non-parametric local regression method introduced by Cleveland in 1979 [Cle79]. This method combines linear least square error regression as well as nonlinear regression. At each point, a low degree polynomial is fitted using only the k-nearest neighbors of the estimated point. The polynomial is fitted using a weighted least square error optimizer: higher weights are attributed to points closer to the estimated point, and lower weights for those further away. The algorithm stops after estimating each

point of the training set with polynomial regression. Different parameters interfere in the estimation: the degree of the polynomial, the attributed weights as well as the percentage of neighbors to use. Note that this method can be applied in $\mathbb{R}^p$ for $p \geq 1$ but in this thesis, we will only present the algorithm in $\mathbb{R}^1$ (*i.e.* with only one variable). The purpose is then to predict $Y = (y_1, \ldots, y_n)$ in function of one variable $X = (x_1, \ldots, x_n)$.
First, we will start by defining our model by the Equation (1.11)

$$y_i = f(x_i) + \epsilon_i \qquad\qquad i = 1, \ldots, n \tag{1.11}$$

where f(X) is an unknown function with no assumptions to be estimated [Cle79]. The errors ϵ_i are supposed independent identically distributed with zero mean and constant variance σ^2 (*i.e.* same hypothesis as in linear model).

Before presenting the implicit equation of the estimation of $f(x_i)$, we should define some parameters and notations. First, one should choose the degree of the polynomial to be fitted locally. In general, we use first or second-degree regression (linear or quadratic regression). A zero degree polynomial is a weighted moving average [Cle79].
Once the degree of the polynomial is set, the subset of neighbors should be selected. For that a parameter α is set as the percentage of k-neighbors with $\alpha \in [0, 1]$. For a point x_0 the model is fitted on $n\alpha$ individuals.

Last, a weight function is defined to privileges the points nearest to the estimated ones. Different weight functions exist in literature, but in general (as well as in R packages), tri-cube weight function is used. If x_0 is the estimated point, each x selected in the neighbor of x_0 is weighted by w(x) defined as:

$$w(x) = \left(1 - |x - x_0|^3\right)^3 \tag{1.12}$$

For each point x_0, let I be the set of $n\alpha$ nearest neighbors. The estimation of the coefficient $\hat{\beta}(x_0)$ is :

$$\hat{\beta}(x_0) = \underset{\hat{\beta}(x_0)}{argmin} \sum_{i \in I} w(x_i) \left(y_i - f(x_i)\right)^2 \tag{1.13}$$

where $f(x_i)$ is either a constant, linear or quadratic function.

1.2.3 Segmented models

For the regression models presented above, the response variable is modeled on the overall range of the predictive variables. Sometimes it may happen that the relationship between the response and some explanatory variables shows a few values where the effect on the response changes abruptly. These values are called breakpoints. Their are different models of segmented regression (logistic ([PG98],[Cox87]), Generalized Linear Models [MN89], ...). In this thesis, we will only present linear segmented regressions: Piecewise regression and Hockey stick regression (Subsections 1.2.3.1, 1.2.3.2) applied in $\mathbb{R}^1$ and MARS (Subsections 1.2.3.3) applied in $\mathbb{R}^p$.

1.2.3.1 Piecewise regression

This is a linear model where the effect on the response changes before and after the breakpoints with continuous regression functions. Piecewise regression may have one or more breakpoint, but in this work, we only consider the case where we have one breakpoint (2 ranges of observations) and two regression models, one for each range. One should estimate not only the regression models coefficients but also the breakpoint.

Breakpoints are non-linear parameters, and standard maximization procedures cannot be used [SW89]. Different scientists were interested in this problem and presented different approaches to estimate the parameters of a piecewise regression ([EF76], [TZ81]). These methods showed significant results, but their estimation of regression coefficients assumes prior knowledge of the breakpoint. In this thesis, we use the approach presented by Muggeo in 2003 [Mug03] (Corresponding to the used R package Segmented [M+08]). Muggeo introduces a linear piecewise regression that can be used for simple or multiple regression as well as with one or numerous breakpoints (not explained in here). No prior information is needed for the estimation other than the starting value of the breakpoint.
Let $Y_{[n\times 1]}$ be the response variable and $X_{[n\times 1]}$ one predictive variable.

The piecewise regression model is presented by Muggeo as:

$$Y = \beta_1 X + \beta_2 (X - t)_+ + \epsilon \tag{1.14}$$

where t is the breakpoint, $(X-t)_+ = (X-t) \times I(X > t)$ with $I(A) = 1$ if A is true. In this equation β_1 is the slope of left line segment (for $X \leq t$) and β_2 is the difference-in-slopes which means that $(\beta_1 + \beta_2)$ is the slope of the right line segment. ϵ is the residual error. The aim is to estimate model parameters (β_1, β_2 and t) in Equation (1.14). Starting with an initial value of the breakpoint, noted $\tilde{t}$, the approach fit iteratively the linear model in Equation (1.15) using the Maximum Likelihood (ML) approach [A+97]:

$$\beta_1 X + \beta_2 (X - \tilde{t})_+ + \gamma I(X > \tilde{t})^- \tag{1.15}$$

where $I(A)^- = -I(A)$ and γ is the parameter known as a re-parametrization of t and accounts for the estimation of the breakpoint t. The iterative algorithm used to estimate parameters in Equation (1.15) is presented below:

1. Fix a start value for the breakpoint, say $\tilde{t}$ and calculate:

$$\tilde{U} = (X - \tilde{t})_+ \quad \text{and} \quad \tilde{V} = -I(W > \tilde{t}) \tag{1.16}$$

2. Fit the linear model with additional variate $\tilde{U}$ and $\tilde{V}$ presented as:

$$\beta_1 X + \beta_2 \tilde{U} + \gamma \tilde{V} \tag{1.17}$$

3. Improve the break-point estimate by:

$$\hat{t} = \tilde{t} + \frac{\hat{\gamma}}{\hat{\beta}_2} \tag{1.18}$$

4. Repeat 2 and 3 until convergence.

When the algorithm converges, $\hat{\gamma}$ is expected to be approximately zero, or rather non-statistically different from zero. The estimators obtained at the final iteration (when the algorithm converge) are assumed the be the ML estimates. In general, the convergence of the algorithm indicates that significant breakpoints are believed to exist. If the algorithm fails to converge, one could not say that there is no breakpoint. A breakpoint could exist but not detected, in this case, the parameters will be estimated as a linear model.

1.2.3.2 Hockey stick regression

Hockey stick regression [HCN76] is a model based on segmentation. This is a special case of piecewise regression where one of the regions is estimated as a constant. A hockey stick regression has two different types of equations type I and type II. In type I, observations lower the breakpoint are estimated as constant (Figure 1.12 a); on the other side, in Type II, the constant model is estimated by the observation higher than the breaking point (Figure 1.12 b). The type I and type II hockey stick are presented in Equation (1.19):

$$\underset{\text{type I}}{f(X) = \begin{cases} \beta_0 & \text{if X} \leqslant \text{t} \\ \beta_1 + \beta_2 X & \text{if X} > \text{t} \end{cases}} \quad Or \quad \underset{\text{type II}}{f(X) = \begin{cases} \beta_1 + \beta_2 X & \text{if X} \leqslant \text{t} \\ \beta_0 & \text{if X} > \text{t} \end{cases}} \tag{1.19}$$

where β_1 and β_2 are regression coefficients and β_0 the constant model to be estimated (see later). Parameter t is the breakpoint to be selected by minimizing the mean square error. This method is general applied in $\mathbb{R}^1$.

Figure 1.12 illustrates an example of simulated data in the form of the type I and type II hockey stick regression.

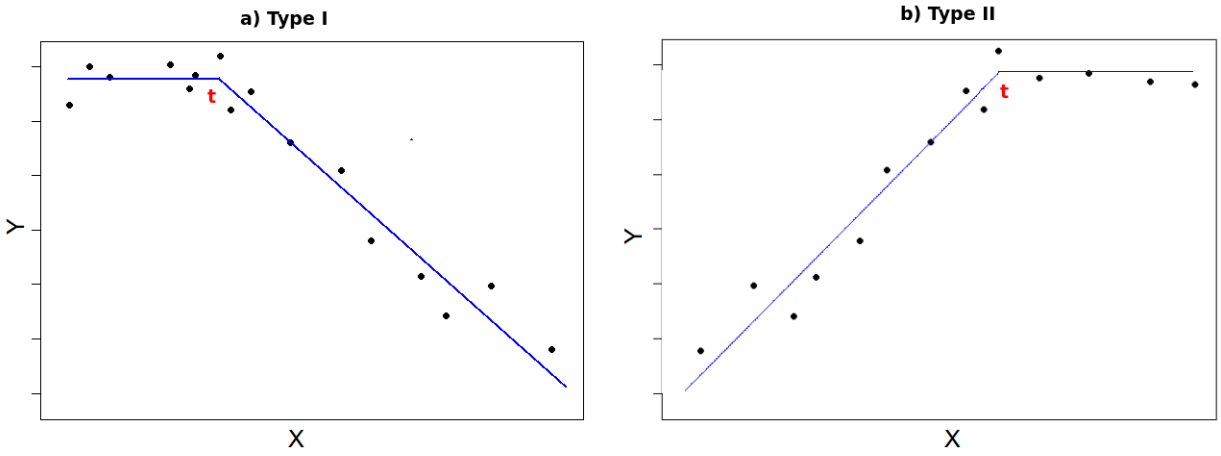


Figure 1.12: ***Hockey stick regression*** *on simulated data (x,y). a) Type I. b) Type II*

The equations presented in Equation (1.19) are not continuous for every β_0, β_1 and β_2. To conserve the continuity of the stick, we decided to proceed as:

- Estimate $\hat{\beta}_1$ and $\hat{\beta}_2$ on the correspondent region using the least square error optimizer
- Calculate $\hat{\beta}_0 = \hat{\beta}_1 + \hat{\beta}_2 \times t$

Other approaches have been used for estimating the coefficients. In particular, Hasselblad [HCN76] proposed to calculate β_0 as the mean of the data and use it as the constant in the linear equation. Notice that one should reconsider the coefficients of the linear equation in this case. Besides, there is one less coefficient to estimate (one less degree of freedom) compared to other methods.
As for piecewise regression in Section 1.2.3.1, the value of the breakpoint is selected by a grid search. First, a set of breakpoints is defined, and data is divided into training and test subsets for model selection. For each breakpoint, coefficients for both types of models (I) and (II) are estimated on the training set. Mean square errors are calculated on the test set. The type of model and the breakpoint with the minimal mean square error on the test subset is selected to fit the data.

1.2.3.3 Multivariate adaptive regression splines

Multivariate adaptive regression splines (MARS) was introduced by Friedman in 1991 [Fri91] as a non parametric regression. The model is similar to a stepwise linear regression, fitted not directly on the original variables but on a new set of transformed variables (defined in Equation (1.21)). In addition this method consider significant interactions between predictive variables (as products of the transformed variables). The model equation is defined by:

$$f(X) = \beta_0 + \sum_{m=1}^{M} \beta_m h_m(X) \tag{1.20}$$

where β_0 and β_m are coefficients estimated by minimizing the residual sum-of-squares, M is the number of all possible functions (see Equation (1.21)) and $h_m(X)$ is a transformed variable or the product of 2 or more transformed variables.

From original variables, MARS creates a new set of variables **C** of dimension $2np$ where n is the number of examples and p is the number of variables:

$$C = \{(X_j - t_{ij})_+, (t_{ij} - X_j)_+\} \qquad j = 1, 2, \ldots, p \quad , i = 1 \ldots, n \tag{1.21}$$

where X_j is the j^{th} variable and $t_{ij} = \{x_{ij}\}_{\{i=1,\ldots,n\}}$, is the observed values of the variable X_j for each individual i. For one variable X_j, and for t (a value of t_{ij}), $(X_j - t)_+$ and $(t - X_j)_+$ are two basis function defined as:

$$(X - t)_+ = \begin{cases} X - t & \text{if X > t} \\ 0 & \text{otherwise} \end{cases} \qquad \text{and} \qquad (t - X)_+ = \begin{cases} t - X & \text{if X < t} \\ 0 & \text{otherwise} \end{cases}$$

Figure 1.13 illustrates an example of the basis functions $(X - t)_+$ and $(t - X)_+$.

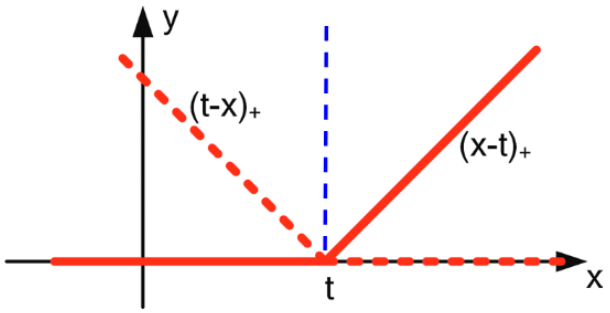


Figure 1.13: ***The basis functions*** $(t - X)_+$ *(broken red) and* $(X - t)_+$ *(solid red) used by MARS to create the new set of variables.*

As well as for hockey stick regression, data is divided into training and test subset. Using Equation (1.20), the first step is to run a forward selection where all possible functions are tested one by one. The function that induces the highest error decrease on the test subset at each time is selected. The model obtained using this procedure overfits the data. This is why the second step consists in running a backward deletion procedure. The term that causes the lowest error increase on the test set is removed. To be noted that the model, when adding interactions, is built with a hierarchical forward step; in other terms, a product of functions can only be tested if at least one of the functions is already selected in the model.

1.3 Artificial Neural Networks

Artificial neural networks (ANN) were inspired from the brain network by McCulloch & Pitts in 1943 [MP43]. Figure 1.14 shows the similarity between the brain and the mathematical neurons. The mathematical neurons take an input vector $\mathbf{x}$ (dendrites in the brain), to compute an output (terminals axons) using an activation function (axons) and an internal function (cell body).

Artificial networks are often known as black box procedures which can learn complex relationships. The first proposed network was a simple perceptron (see Section 1.3.1) then it was developed later to multilayers perceptrons and deep networks (Section 1.3.2) especially convolution networks (Section 1.3.5).

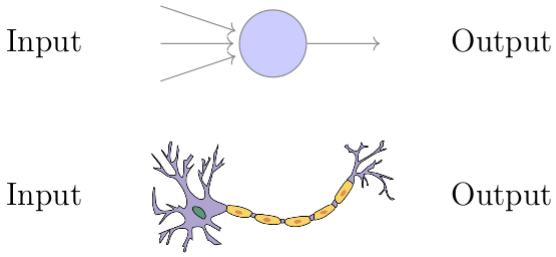


Figure 1.14: ***A brain neuron Versus a mathematical neurone.*** *Picture extracted from "Prostate Cancer Classification using Convolutional Neural Networks, Anna Gummeson 2016"*

1.3.1 Simple perceptron

Simple Perceptron is the first form of artificial neural network. Introduced by Rosenblatt in 1958 [Ros58], it consists of one binary neuron, *i.e.* one output y which is 0 or 1. An illustration of one perceptron is presented in Figure 1.15.

At each input $(x_1, \ldots, x_n)$ a weight $w_1, \ldots, w_n$ is assigned. The output y is estimated by an activation function f applied on an intermediate output ξ calculated by a summation function with a bias θ.

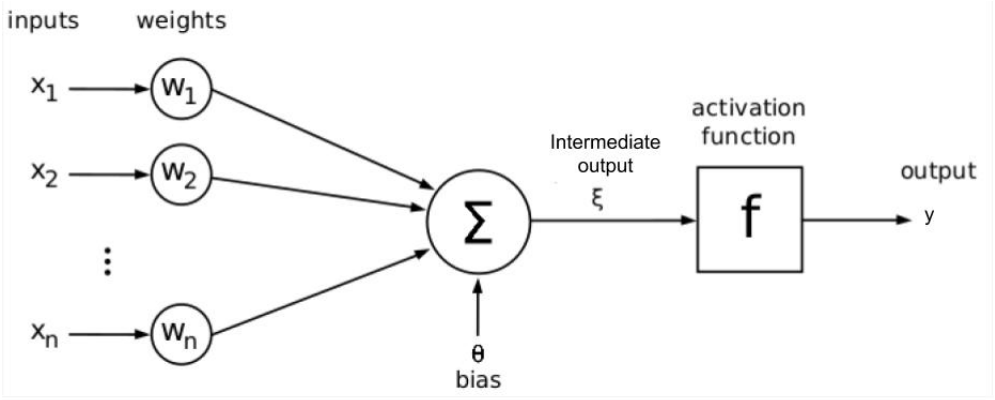


Figure 1.15: ***Illustration of a simple perceptron*** *with n inputs, one output and an activation function f*

The intermediate output ξ of the neuron is computed as:

$$\xi = \sum_{i=1}^{n}(w_i x_i) + \theta \tag{1.22}$$

Then the output y is computed as:

$$y = f(\xi) = f(\sum_{i=1}^{n}(w_i x_i) + \theta) \tag{1.23}$$

with f the step activation function defined as:

$$f(x) = \begin{cases} x & \text{if x} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{1.24}$$

A network of two perceptrons with three inputs would look like the one presented in Figure 1.16. Note that they do not interact with each other. We call this a "single layer perceptron network".

A simple perceptron (only one neuron) could be seen as a classifier, that can discriminate only two classes with a straight line. This method is not able to solve XOR (Exclusive OR) function. This function is not linearly separable. Lets consider the 2 dimension problem presented in Figure 1.17 where data should be classified in two classes: (0,0), (1,1) in

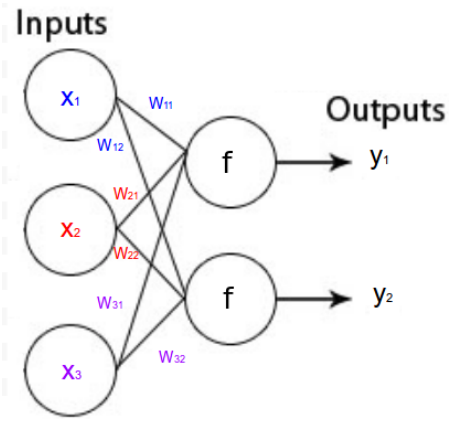


Figure 1.16: ***Two perceptrons network*** *with three inputs* (x_1, x_2, x_3) *and two outputs* (y_1, y_2)

one class and (0,1), (1,0) in the other class. The perceptron will fail to find the line that separates these data [RN16].

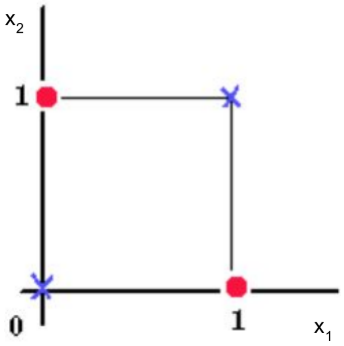


Figure 1.17: ***Set of examples that are not linearly separable*** $\mathbb{R}^2$. x_1 *and* x_2 *are the perceptron inputs. The first class is represented by a blue cross and the second by a red circle*

1.3.2 Multilayer neural networks

The multilayer network is a feed-forward network with one or more hidden layers, of which the computation nodes are called hidden neurons or hidden units. The term hidden is used because those layers are seen from neither the input nor the output layers. The task of these hidden units is to be part of the analysis of data flowing between the input and output layers. By adding one or more hidden layer the network, higher order statistics may be extracted from its input. The feed-forward term comes from the fact that the

information flows only from the input layer through the hidden layer(s) to the output.

1.3.2.1 Multilayer perceptrons

Multilayer perceptron (MLP) [RHW85] is the most popular class of multilayer feed-forward ANNs and was developed to overcome the limits of a simple perceptron. MLP consists on stacked hidden fully-connected layers with specific activation function (see Section Activation functions) to learn non-linear and linear relationships between input and output layers. In fully-connected layers, each neuron is connected to each other neuron in the previous and the next layer. Training MLPs involves adjusting the parameters (weights and biases) of the model by Backpropagation (see Section 1.3.3.2) to minimize a cost function.

Three layers MLP:

Considering an example of MLP with: i) an input layer (x_1, x_2, and x_3), ii) one hidden layer with 4 neurons, an activation function f, a weight matrix W and a bias vector θ and iii) an output layer with response variable Y. Each layer is fully connected to the previous and next layers. The example is presented in Figure 1.18.

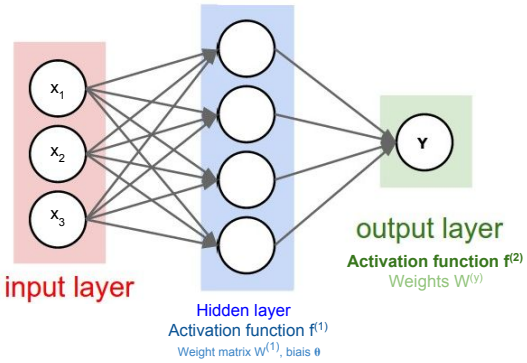


Figure 1.18: ***Multilayer perceptron*** *with 3 layers where the first layer is an input layer (x_1, x_2 and x_3). The last layer is the output layer with one response variable Y and an activation function $f^{(2)}$. These layers are fully connected to a hidden layer with 4 neurons. The hidden layer is characterized by an activation function $f^{(1)}$, weight matrix W, and bias vector θ. Figure adapted from* `http://blog.christianperone.com`

Similar to the perceptron, the mathematical formula of the MLP presented in Figure 1.18

can be written in two equations, one corresponding to the intermediate output ξ of the hidden layer, and one given for prediction Y. The intermediate output is calculated as:

$$\xi_j = f^{(1)}\left(\sum_i (W_{ij}{}^{(1)} x_i) + \theta\right) \tag{1.25}$$

The prediction of the response variable is computed as:

$$Y = f^{(2)}(\xi) = f^{(2)}\left(\sum_j W_j{}^{(y)} \xi_j\right) \tag{1.26}$$

The reasoning is similar when adding more hidden layers with specific activation function and weights matrix. Model architecture is critical and should be chosen with caution (see Section 1.3.6).

1.3.2.2 Activation functions

Each hidden layer and the output layer of a multilayer network is characterized by an activation function. The activation functions most commonly used for perceptrons are the following (see Figure 1.19).

1. **Linear function:** This function is only used on the output layer for regression problems:

$$f(x) = \beta x$$

2. **hyperbolic tangent:** Used in general for hidden layers. It works for output values in interval [-1,1]:

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

3. **Sigmoid function:** This logistic function is used on the output layer for classification problems (output in [0,1]):

$$f(x) = \frac{1}{1+e^{-x}}$$

4. **ReLU:** Rectified linear unit's function is used for hidden layers. This function benefits from its simplicity, resulting in faster training [KSH12]:

$$f(x) = max(0, x)$$

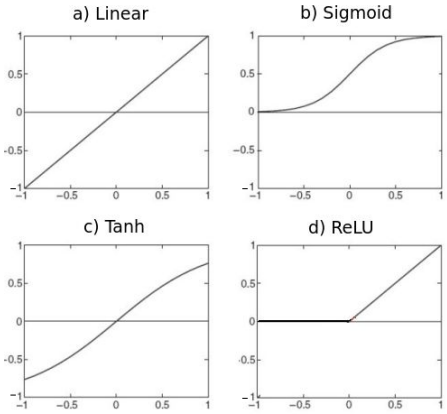


Figure 1.19: ***Visual comparison of different activation functions.*** *a) linear. b) Sigmoid. c) hyperbolic tangent. d) Rectified linear unit*

1.3.2.3 Regression and classification

Multilayer networks are suitable to solve regression as well as classification problems. The difference between those tasks is in how the output layer is presented: continuous response for regression and binary response for classification. This difference induces a difference in the activation function of the output layer (see Section 1.3.2.2). Besides, in the training process, the difference will be seen in the definition of the cost function that will be minimized (see Section 1.3.3).

1.3.3 Training of a neural network

The process aims to find the optimal parameters of the network. Training DNNs is an iterative process where the outputs created on each input are analyzed, and the network (parameters) is repeatedly being adjusted to produce better results. The network is considered to be trained when minimizing a define cost function (presented below). In this

thesis, we are interested in supervised learning where the dataset is labeled. Such dataset consists of input patterns for the network with their corresponding labels - observed network outputs.

1.3.3.1 Cost function

As mentioned in the previous paragraph, during the training, the algorithm minimizes a cost function that should be defined. This function is used to measure the performance of the network. The cost function represents the differences between predicted and observed values. In the regression case, that is typically the mean square error. It corresponds to the minimization of the Gaussian likelihood with expectation equal to the function defined by the network. In classification, the cost function can be the error rate, that corresponds to minimization the cross-entropy. The cost function for a neural network is similar to that used to optimize parametric models. The principle of maximum likelihood is often used, and the cost function is the negative log-likelihood which is the cross-entropy between training data and model predictions. In regression models, the cross-entropy is equivalent to mean square error.

1.3.3.2 Gradient-descent algorithm

The training algorithm of a neural network is based on gradient descent optimizer [Cau47] applied to minimizes the cost function by updating the parameters. This is an iterative algorithm, and on each iteration, parameters are updated in the opposite direction of the gradient of the cost function considering the parameters. The size of the step we take on each iteration to reach the local minimum is determined by the learning rate η.
The steps of a gradient-descent optimizer, for a defined learning rate η, are as following:

1. Initialize the network parameters (weights and bias).
2. Calculate the derivative (gradient) of the cost function. The derivative refers to the slope of the function at a given point that defines the direction for updating weights. The gradient (*i.e.* change in the cost function) when the parameters are changed by a very small value η from their original value.

3. Adjust each parameter using the gradient descent updates with learning rate η.

4. Repeat steps 2 and 3 until further adjustments to weights do not significantly reduce the cost function.

Gradient-descent optimization algorithms The choice of the learning rate η is critical. A very low learning rate leads to slow convergence, while a very high learning rate can lead to divergence. Besides, a constant learning rate through the process is not ideal. For these reasons, different optimizers of gradient descent were implemented to overcome the limitations of the choice of η [Rud16]. In this thesis, we present two optimizers: RMSprop and ADAM. For these two optimizers, an initialization of the learning rate is requested. In general it is in powers of 10, specifically $0.001, 0.01, 0.1, 1$.

1. Root Mean Square Propagation (RMSprop) is not published yet, and it was presented by Geoff Hinton [HSS]. RMSProp divides the overall learning rate by the square root of the error of the previously updated gradients for a given parameter. The updated gradients are exponentially weighted by a moving average. This means that old values contribute less than new ones.

2. Adaptive Moment Estimation (Adam) [KB14] computes adaptive learning rates for each parameter. Adam is an extension of RMSprop with a bias correction using the mean and the variance of the past gradients for parameter updates. Using Adam, the learning rate will decrease, as it approaches the minimum.

Overall, RMSprop and Adam lead to similar results in most of the cases with a slight outperformance of Adam in certain networks [Rud16]. The choice of the optimizer is made with respect to the data and the architecture.

Gradient-descent solution: backpropagation algorithm

Backpropagation [RHW86] is the algorithm used in machine learning to calculate the gradient (derivative) of the cost function needed to update the neural network parameters

(weights and bias). The result of this algorithm is a neural network configured to minimize the cost function when solving a given problem using a gradient descent algorithm. The algorithm consists of two steps: forward and backward propagation. In the forward propagation, from the inputs, using the weights and the activation function, we compute the output of the network. One should notice that it is essential to initialize weights and the biases of each layer (see Section 1.3.6). The backward propagation is used to adjust the weights of neurons in function of the learning rate η and the descent gradient of the cost function. This is an iterative algorithm. An epoch (iteration) is defined as a forward and backward propagation. The number of epoch should be set. The backpropagation algorithm is described in Algorithm 3.

Algorithm 3: Backpropagation algorithm

1. Propagate the input through the network layer and calculate the predicted output of the model (Forward propagation).
2. Calculate the cost function between predicted and observed output.
3. Backpropagate the cost function from the output layer to the input layer passing by all hidden layers
4. Calculate the derivatives of the cost function at each node for the output layers as well as for each hidden layer.
5. Calculate the update rule using the selected optimizer in function of the derivatives and the learning rate η
6. Update weights using the following equation:

$$\text{New weight} = \text{old weight} + \text{update rule} \tag{1.27}$$

7. Repeat until convergence or until reaching the defined number of epochs
8. **return** Trained optimized network and weights at each layer (w_{ij}^k).

1.3.3.3 Overfitting

A common problem in the learning process is overfitting. One method to avoid overfitting in training neural networks is to split the dataset into a training and a validation set.

A good starting point for determining this ratio is to put 80% of the available data into the training set and 20% into the validation set. While learning, the performance of the network is regularly examined on the validation data set by the cost function. Figure 1.20 shows the evolution of the cost function (error) on the training and the validation set. While going towards the right of the figure (forward in the learning process), the complexity of the model increases such that the training error is reduced, but the validation error does not. This is when we speak about overfitting. When the cost function on the validation data reaches a minimum and start to increase (see Figure 1.20) the network is considered trained and can be generalized to other sets of data. The minimum is known as the stopping point at which the model with the best parameter estimators is returned.

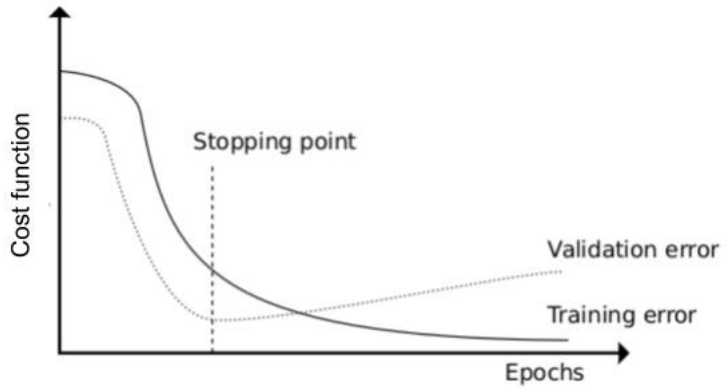


Figure 1.20: ***Evolution of the cost function*** *in the training dataset Vs. in the validation dataset. The stopping point is the minimum of the cost function on the validation dataset. Figure adapted from [Voj16]*

In general, the training continues until reaching the defined number of iterations even after hitting the minimum point on the validation set. To avoid the high computational time and to prevent additional overfitting, early stopping concept was introduced. Early stopping [Pre98] is a technique for controlling overfitting by stopping training when the performance has stopped improving on a validation set after a number of iterations. A parameter known as patience (pa) should be identified for early stopping. The patience (pa) is the number of epochs with no improvement after which training will be stopped. After (pa) iterations, if the cost function on the validation set is higher than it was the

last time it was checked (the stopping point in Figure 1.20, the training is stopped. The model with the lowest cost function is retained.

Mini-batch gradient-descent

Gradient descent can vary in terms of the number of training data used to calculate the cost function, that is in turn used to update the network. The gradient descent has three main types: batch, stochastic, and mini-batch [Rud16]. In this thesis, we use the mini-batch gradient. Mini-batch gradient descent provides a balanced optimization between the robustness and the efficiency of other gradient descent and helps to overcome overfitting. The algorithm splits the training dataset into small batches that are used to calculate the cost function and update network parameters. Even though mini-batch requires an additional parameter (the mini-batch size), but it shows several advantages: it is faster than batch and stochastic gradient because of the lower number of examples and it reduces the variance of the parameter updates, which can lead to more stable convergence [Rud16]. The choice of the mini-batch size, known in implementation as the batch size, depends in general on the architecture of the network. Common batch sizes vary from 50 and 256 and are, for the Graphics Processing Unit (GPU) memory reasons (see Chapter 4), a power of 2 [Jai17].

Cost function regularization

Computing validation error and early stopping are used to overcome overfitting. However, other methods, known as regularization, used at each layer, or between layers of a network can be used for the same purpose. Regularization is a technique that implies slight modifications to the cost function allowing a better generalization of the model. There are different types of regularizations. In this thesis, we present two different types: weights regularization and dropout.

Norm penalties regularization ℓ_1 and ℓ_2 are the most common types of norm penalties regularization. This regularization modifies the general cost function by adding another

term known as the regularization term applied on network weights.

$$\text{Regularized cost} = \text{cost function} + \text{Regularization term} \tag{1.28}$$

This regularization term forces a penalty on weights and induces a convergence (or equality) of the weights to 0 (see Section 1.2.1.2) for details). This decrease is due to the regularization term that forces. Therefore, it will also reduce overfitting to quite an extent.
The regularization term differs in ℓ_1 and ℓ_2. The definitions are similar to those presented in Section 1.2.1.2 to explain penalized regressions. Each regularization term is composed of a regularization parameter λ to be optimized and $||W||_1$ or $||W||_2$ the norms of the weight matrices using ℓ_1 or ℓ_2 respectively. ℓ_1 is very useful when trying to compress our model. Otherwise, usually ℓ_2 is used.

Dropout [SHK+14] is a hidden layer that can be applied to hidden layers as well as for the input layer. The term dropout refers to dropping out some units which mean temporarily removing it from the network, along with all its incoming and outgoing connections. The choice of which units to drop is random. A dropout layer is characterized only by a fraction p representing the fraction of neurons to be randomly dropped from the target layer. The parameter $p \in [0, 1]$ has to be tuned (see Section 1.3.6).
Figure 1.21 shows for an iteration in the training process, the effect of applying dropout on the input layer as well as on both hidden layers with different fractions p.
Each iteration has a different set of nodes, which results in a different set of outputs, reason why dropout can be viewed as a form of averaging multiple models. Notice that the dropout is applied only in training, and at the end of the processes, all weights of all neurons are estimated. In the validation process, all neurons are considered, and each activation is reduced by a factor p. Dropout roughly doubles the number of iterations required to converge. However, training time for each epoch is less [PY].

1.3.4 Deep learning

Deep learning [GBC16] is a set of learning methods attempting to model data with complex architectures combining different non-linear transformations.

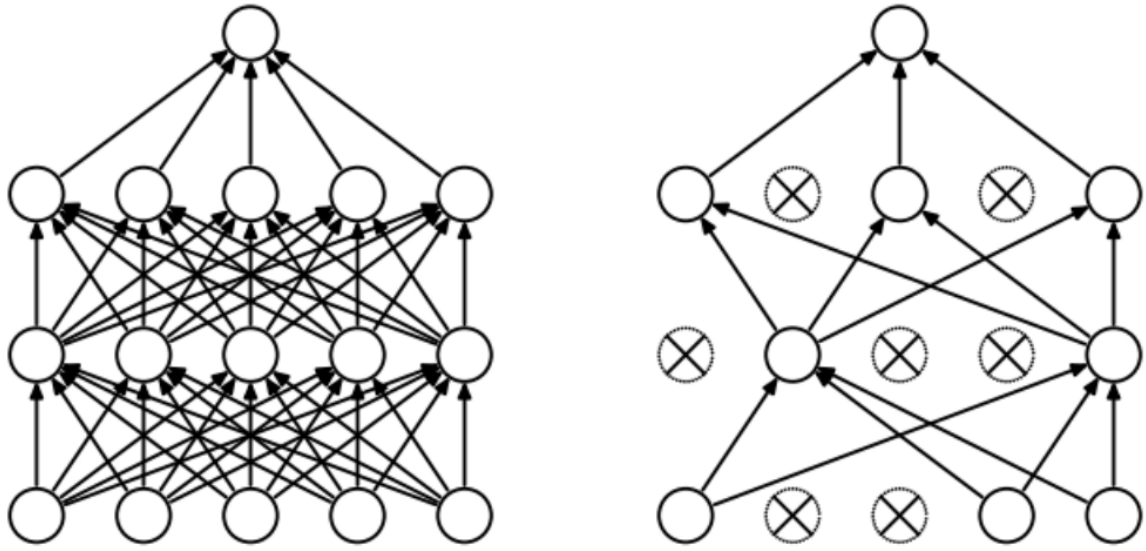


Figure 1.21: ***Dropout in the neural network.*** *Left: A fully connected network with two hidden layers with five neurons each. Right: An example of one iteration in the training of the network on the left with dropout layers applied on the input layer as well as for each hidden layer with fractions 0.4, 0.6, and 0.4 respectively. Crossed neurons have been dropped in this iteration. Figure from [SHK+14]*

The multilayer perceptron is a type of deep learning. When the problem is more complex, and MLP is not sufficient, we start talking about deep feedforward neural networks that have the same architecture as MLPs with a large number of hidden layers (neurons).
Even though for an important time, the term deep referred to a high number of layers and of neurons but then, different other types of networks were developed under the name of deep learning. More precisely, two networks that are widely used in image processing, speech recognition, computer vision, text classification, and others: i) the Convolution Neural Network (CNN) and ii) the Recurrent Neural Network (RNN). For these two types, deep learning does not necessarily mean a larger number of layers, but rather a very high number of neurons. In this thesis, we will focus only on the convolution network.

1.3.5 Convolution networks

Convolutional Neural Networks (CNNs) are deep neural networks that take account of spatial dependence in the input data. They were originally inspired by Hubel and Wiesel's [HW63].

A convolution network is designed to model input data in the form of multidimensional arrays (images) [L+16] or one dimensional sequence (genomic or text) ([ADWF15], [SS09]). The size of these data makes a fully connected neural network very challenging since the number of parameters in such models would exceed the number of training data. Convolution networks add assumptions on the structure of the network to force only local connections (not fully) and reduce the effective number of parameters to learn. Contrarily, the convolution network has significantly high computational cost. If the network is pretty deep, each training step is going to take much longer. This limit is becoming less questionable with the development of GPU's. Besides, the training of such a network needs many data.

Convolution networks are based on a stack of convolution, and pooling layers, followed by fully connected layers used to predict the output (see Figure 1.23). Convolution can be used for different input dimensions, 1D, 2D and even higher. In this thesis, we will only present the 1-dimensional convolution network (for genomic sequences).

In a Convolution Neural Network, the hidden layers may have different types. Most of these layers are optional, but there has to be at least one convolution layer for the net to be called a convolution network. In the below sections, we will present each layer that can be used with this type of networks as well as the choice of architecture and training process.

1.3.5.1 The input layer

The input layer represents the entries of the model (predictive variables). It is the first layer in the network, and as such, it is not counted to the number of CNN layers. For 1D convolution, the input layer is a set of matrices where each matrix contains predictive variables of one individual. For example, when studying DNA sequences using a convolu-

tion network, the input layer is a matrix with a dimension $(4 \times l)$ where 4 is the number of DNA bases and l is the length of the DNA sequence. The dimension of the input data defines the dimension of the neurons in the first convolution layer (see Chapter 4).

1.3.5.2 Convolution layers

Convolution is a mathematical operation which takes two functions q and g as input and generates a function which is the summation of multiplication of q and a reversed and shifted version of g. Formally, the convolution operator (*) is defined as in Equation (1.29):

$$(q * g)(x) = \sum_{t} q(x)g(x - t) \tag{1.29}$$

In convolution networks, g is known as the convolution filter and acts on the input q. The resulting output is called a feature map. In general, the input function is a multidimensional array (see Section 1.3.5.1).

A convolution layer plays the role of a scanner that will detect the presence of certain features in the input data. It consists of multiple filters, each recognizing certain specific features. A filter is a multidimensional array of parameters, consisting of the weights of the connections and the bias. All filters in the same layer have the same length. In 1D convolution, a filter is a matrix of dimension $w \times N$, where w is the length of the filter and N is the number of rows of the input X (For example $N = 4$ with genetic sequences (see Section 4.4.1.1). A filter K scans an input X, and the result is then calculated as the sum of the product between the filter K and the relevant portion I of the input X. More precisely, the output of the filter at a position j is calculated as in Equation (1.30):

$$f\left(\sum_{k=1}^{w}\sum_{n=1}^{N}(I_{n,k}K_{n,k}) + \theta\right) \tag{1.30}$$

where f is the activation function of the convolution layer. Figure 1.22 shows the output of a convolution with an input X and a filter K with $N = 4$ and $w = 4$. At each position j, the filter scans a region I_j and calculates the matrix product to produce the feature map. The blue box in the feature map is the output for region I_1 while the green box is the output for region I_3.

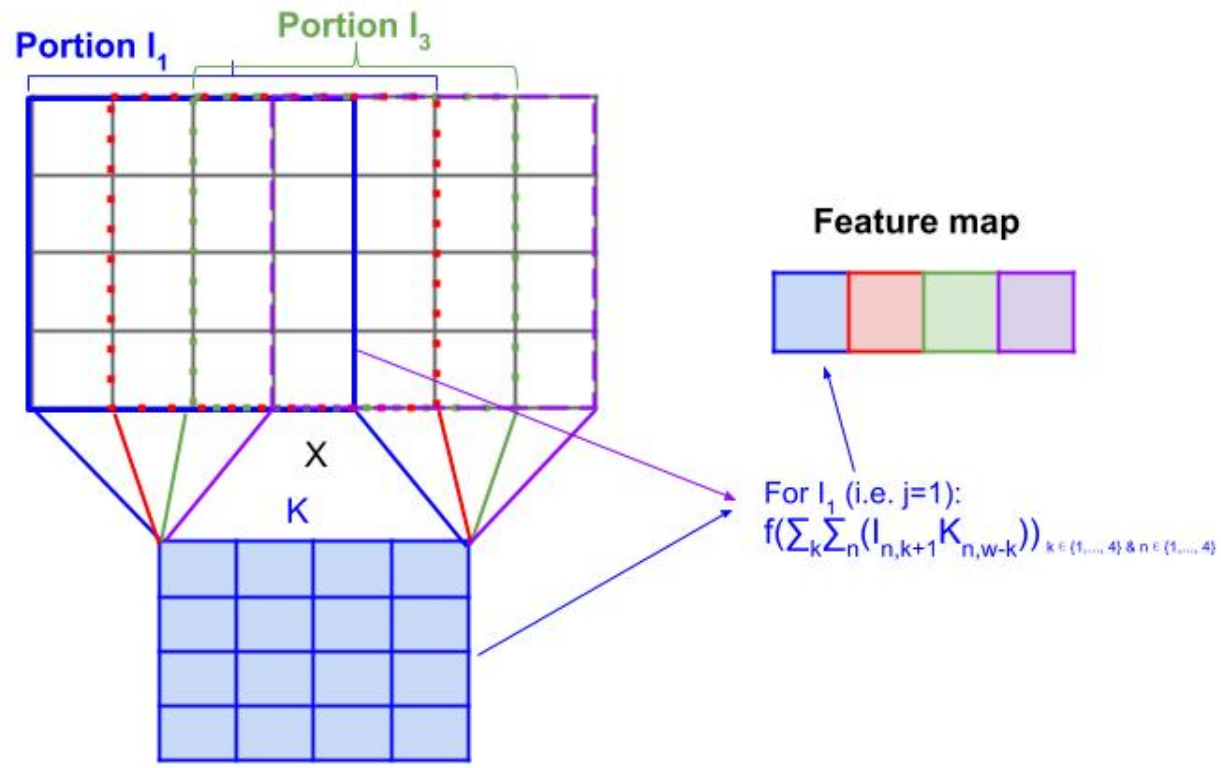


Figure 1.22: ***Convolution layer*** *with an input X and a filter K. The output (feature map) of the convolution layer function calculated by Equation (1.30) is presented on the right.*

In contrast to traditional multilayer neural networks, where higher level neurons are connected to all previous neurons, the convolution units are only connected to a local region in the input. This sparse connectivity allows the CNN to exploit the local correlations efficiently. The activation function of this layer is usually a ReLU. At the end of a convolution layer, for each filter, the weight matrix is estimated. Higher weights mean a higher occurrence of a filter in the input. The output of this layer has the same spatial form of the input with reduced dimension.

During training, one can regulate overfitting in convolution layers by adding an ℓ_1 or ℓ_2 regularization. Dropout is rarely applied to this type of layers especially when they have followed the input layer.

1.3.5.3 Pooling layers

The pooling layer can be used to compress spatial information of the feature maps. In particular, the pooling layer makes the network less sensitive to small changes in the location of a feature, *i.e.* the output of the pooling layer remains the same even when a feature is moved a little. Pooling also reduces the size of the output of the convolution layer, which simplifies computation in later layers.
There are different pooling operations, but the most used in practice is the maximum pooling. To perform max pooling, we slide a window across the output of the convolution layer. As the window moves across the sequence, the window is resumed by the largest value. An average pooling also exists where for each window the mean is calculated.
The size of the window is a parameter to be tuned. There is no specific rule to set the size of the pooling window, in general, it is tuned by hand from prior knowledge.
Convolution/pooling layers can be stacked onto each other, neurons in higher layers will cover increasingly larger parts of the input. This is valid in 2D and 3D convolutions. In general, in 1D convolution, the model architecture contains one convolution/pooling layer followed by one or more fully-connected layers.

1.3.5.4 Fully-connected layers

As presented before, the convolution layers are used to extract features from the data. After feature extraction, fully-connected layers are used to learn a function that predicts the response variable. These layers are also known as dense layers. The last layer of a CNN network is a fully-connected layer with a specific activation function: i) sigmoid for classification or ii) linear for regression. Also, the number of neurons in this layer is similar to the output layer: i) number of classes in a classification problem and ii) number of outputs in regression (*e.g* 1 if one-dimensional regression) .
In addition to this final layer, a stack of dense layers with a ReLU activation can be added to the network and increases the non-linearity aspect. The number of neurons in each dense layer is set in function of the network and have to be tuned by experience. To avoid overfitting, each of these dense layers can be regularized by either ℓ_1 and ℓ_2 regularization or by adding after each layer a dropout layer with a specific fraction p.

Figure 1.23 shows an example of a convolution network with one convolution/pooling layer and one fully connected layer. The example shows an input matrix X used to predict the output layer Y. The convolution layer contains three filters (boxes in color) of dimension 4×4. The output of the convolution layer is then reduced using a max-pooling layer with a window size equal to two. The last layer is a fully connected layer with four neurons.

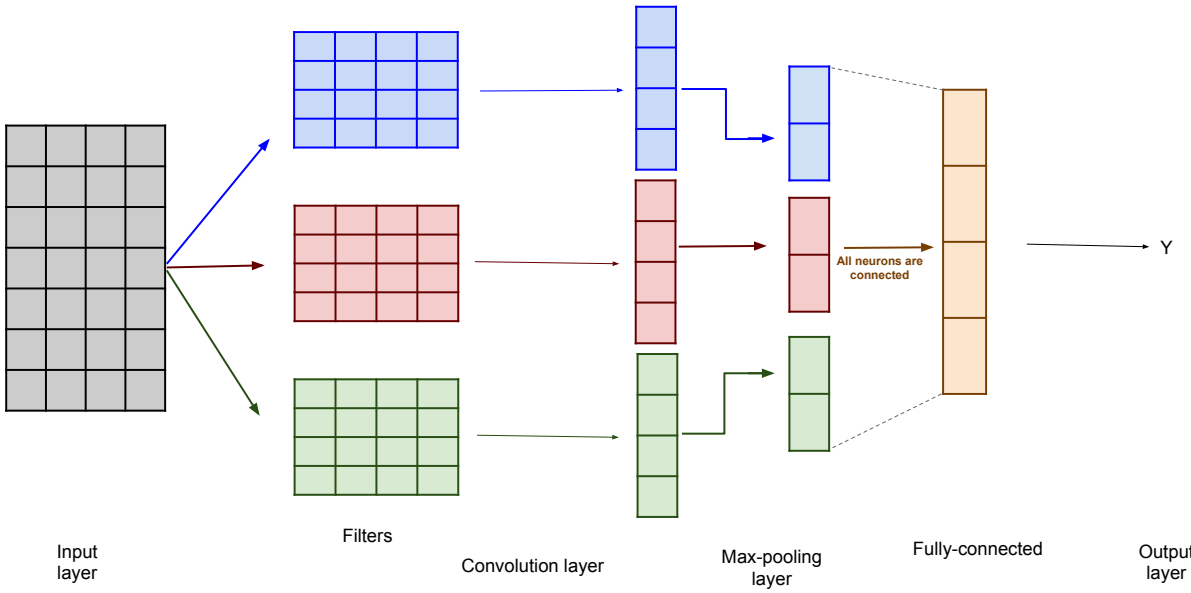


Figure 1.23: ***Convolution neural network*** *with one convolution/pooling layer and a fully-connected layer. Figure inspired from [AWLS16]*

1.3.5.5 Training

Convolution network is one type of DNNs, and it is trained using gradient descent algorithm as well (see Section 1.3.3). Model parameters (weights and bias) are estimated by the descent gradient using the backpropagation algorithm. The learning rate and the optimizer must be carefully chosen by hand (see Section 1.3.3). Also, during training, one can regulate overfitting in adding ℓ_1/ℓ_2 regularization or dropout layers. To be noted that dropout is rarely applied after convolution layers especially when they followed the input layer.

1.3.6 Tuning network parameters

For the different models of artificial neural networks, MLP, and CNN, several parameters are to be initialized and chosen to define the architecture of the network. In Section 1.3.3, we already discussed the choice of the learning rate η, the batch size, and the optimizer. In this section, we present other parameters, their sets of choices as well as their effect on the network.

Before presenting the different parameters, one should mention that there are no defined strategies to choose each parameter, they are dataset dependent and should be chosen by a grid search over different parameters: For a parameter P, all other parameters are fixed, and for different values of P, the model is fitted and evaluated (using cross-validation). The final parameters are the ones that minimize the cost function on the validation set.

Here are the different parameters to be tuned:

1. The number of hidden layers and the number of neurons per layer: these parameters depend on the number of input and output data as well as on the training algorithm and the activation function. In general, we start by a network with a few numbers of layers and increase the number of hidden layers and the number of neurons successively until no additional improvement is seen. Too many hidden units will make the training unnecessarily slow and will result in poor generalization.

2. The initialization of weights and bias: the process of gradient descent requires small (even zero) weight and biases initialization. The bias can be initialized to zero with no effect on the backpropagation algorithm. However, setting all weights to zero will induce symmetry to the algorithm $i.e$ the layers will receive the same gradient, hence performing the same update and remaining identical, thus wasting capacity. Provide that, weights are generally initialized to uniform random small values and bias to zero [Ben12]. Furthermore, known weight matrices may be used for initializing depending on the model hypotheses (always positive, pre-defined weights). In general, the final network performance is independent of the choice of initial weights, but this needs to be checked on the data.

3. Dropout rate: Values between 0 and 1 are accepted. Empirically, a rate between 0.4 and 0.6 is the most used rate.

4. Number of Epochs: Try different values based on the time and the computational resources. A high number of epochs may over-fit to the data, and a lower number of epochs may limit the potential of the model. This can be controlled by early stopping criteria where a high number of epochs is chosen, and the model stops earlier if no improvement on performances is detected after a defined number of epochs (patience). Patience is also a parameter to be tuned

5. Regularization: Choice between ℓ_1 and ℓ_2 regularization. Besides, the regularization parameter λ should be set in general to a small value of around 0.001.

1.4 Bioinformatic context

Understanding gene expression regulation and identifying its components is a challenging problem that was studied under a different point of views. Bioinformatics and biostatistics have a huge role in revealing novel and fundamental aspects of the mechanisms of gene regulation. In this section, we present different bioinformatics and biostatistics studies about gene expression linked to our work.
Subsection 1.4.1 describes how gene expression can be predicted from experimental and epigenetic data, while in Subsection 1.4.2 studies about modulation of these epigenetic data by DNA sequences are considered. In the third Subsection, we briefly present the work with a similar objective as ours: predicting gene expression using DNA sequence. In the last subsection, we explain our motivation behind the Chapter 3 when integrating and searching for interactions between variables.

1.4.1 Prediction of gene expression using experimental data

When explaining gene expression in Section 1.1, we presented different biological components that are involved in gene regulation at the transcriptional level, especially transcription factors, DNA methylation, histone modifications, and others. In this section, we

highlight some works that aim at predicting gene expression using transcriptional regulation data. Note that binding of transcriptional regulators can be quantified by experiments using the chromatin immunoprecipitation (ChIP) sequencing technology [JMMW07] which is a powerful method based on ChIP [KA99] for identifying genome-wide DNA binding sites for transcription factors, histone modifications and, other regulators.

In 2009, Ouyang *& al.* proposed a regression method to predict gene expression based on TF-DNA binding quantified by ChIP-seq. First, TF association strength was built using a weighted sum of the ChIP-seq signal of 12 TFs located around the TSS of each gene. Then a two-stage model was used: in the first stage, they used principal component analysis (PCA) to extract uncorrelated combinations of TFBS. At a second stage, a linear model was fitted to predict gene expression using PCA variables. The model was applied using RNA-seq expressions in mouse cells and yielded a Pearson correlation of 0.81 between observed and predicted gene expression. This method used only TF bindings and did not take into consideration histone modifications. However, Karlic *& al.* [KCL+10] presented, in 2010, a similar regression model to predict gene expression as measured by microarray in human CD4 T+ cells using only 38 histone modifications quantified by ChIP-seq. The model also had a high performance with a Pearson correlation of 0.74. Two years after, Mcleay *& al.* [MLCPB12] proposed to predict gene expression using both TF bindings and histone modifications as well as chromatin accessibility data. Based on a regression model and 12 TFBs in [OZW09] 7 histone modifications and chromatin accessibility, the model performances reached a Pearson correlation of 0.835.

Cheng *& al.* [CAM+12] presented some limitations of the approach shown below, in particular, the low number of TFs and the fact that the models were tested in only one cell line. To overcome these limitations, Cheng *& al.* proposed to predict gene expression quantified by CAGE expression (Cap Analysis of Gene Expression [SKK+03]) using the binding site of 120 TFs in the human genome in 6 cell types. They tested four different regression models and restrained their studies to Random Forest that presented the higher accuracy (Pearson correlation = 0.81). Besides, they showed that when adding 12 different histone modifications to predictive variables, the prediction correlation increases to 0.92.

The relationship between gene expression and experimental data (TFs, ...) was also studied in cancer cells. In 2014, Li *& al.* [LLZ14a] presented RACER for Regression Analysis of Combined Expression Regulation to predict mRNA expression (RNA-seq) based on transcriptional and post-transcriptional data in Acute Myeloid Leukemia (AML) with a Lasso penalized linear regression [Tib96]. In addition to 97 TF ChIP-seq signal, they integrated expression of 470 miRNAs, as well as DNA methylation and copy number variation (CNV) of each gene, as two additional predictive variables. Note that TF binding signals were calculated on +/-50 bp around TSS. The model was tested on 173 AML patients. A model including all variables had a median Spearman correlation of 60%. They further showed that CNV and miRNA have a minor contribution to the model performance, and DNA methylation increases the model by only 5%. The highest effect is attributed to TF binding. Later in 2015, Jiang *& al.* proposed RABIT (Regression analysis with background integration) [JFLL15a], another regression framework to predict gene expression differentiation using TF ChIP-seq signals in cancer cells. The originality of this work is the control of the background effect such as the copy number variation (CNV) and DNA methylation. The performances of RABIT showed higher accuracy than other methods (AUC = 0.72). Also, they were able to identify cancer-associated TFs.

Finally, Schmidt *& al.* [SGG+16] presented a new approach to predict gene expression. Provided the limits of ChIP-seq data (necessity of huge amount of biological materials, high costs, ...), they first proposed TEPIC, a segmented method that predicts TF binding by combining position weights matrices [Sto00] and open chromatin regions (OCRs). In a second step, they used the TEPIC TF scores to predict gene expression using an elastic net regression [ZH05]. The model was tested in 4 cell lines. They showed that when predicting gene expression using TEPIC, that is based on one single open-chromatin assay, it is possible to achieve approximately the same accuracy as when using several expensive ChIP-seq assays.

1.4.2 Predicting epigenome marks via the DNA sequence

In Section 1.1.3, we explained the importance of epigenome, especially histone modifications, DNA methylation, and DNA structure in gene regulation at the transcriptional level. It is then important to understand the main causes of these modifications. It was known that the epigenome in a cell is dynamic and can be affected by environmental factors or diseases [MA16]. Along the years, different studies showed that DNA sequences have a major effect on the epigenome.

In 2013, three papers published in Science ([KKPG+13], [KWG+13], [MvdGD+13]) showed a strong relation between DNA sequence variation and histone modifications. Furthermore, McVicker *& al.* [MvdGD+13] developed the "combined haplotype test" and identified QTLs (quantitative trait loci: specific DNA regions) associated with histone modifications.

One of the most exciting works that underline a relationship between histone modifications and DNA sequence was presented in 2015 by Whitaker *& al.* [WCW14]. They created a pipeline called Epigram to predict histone modifications and DNA methylation from DNA motifs. The method succeeded to select motifs that discriminate modified regions with 79% accuracy. Besides, it revealed mark-specific motifs associated with chromatin-modifying enzymes.
Furthermore, Illingworth *& al.* [IGSDS+15] studied alterations in DNA methylation in brain cells. They suggested that DNA sequence compositions are the principal determinant of the human brain DNA methylome. More precisely, they showed that hypomethylated Deferentially Methylated Regions (DMRs) are enriched with CpG islands where, on the other hand, the regions that are CpG-deficient are hypermethylated. Quante and Bird (2016) [QB16] discussed the effect of short, frequent DNA sequence motifs on the epigenome. Short motifs are in general 2-5 bp and thus are more frequent in the DNA sequence (CpG, AT, ...). Quante and Bird proposed that proteins recognizing short, frequent AT - rich sequences, can as well modulate aspects of chromosome structure.

Some scientists also used deep learning to predict epigenetic marks using DNA sequences. Angermueller *& al.* [ALRS17] presented DeepCpG a computational method based on con-

volution neural networks (Section 1.3.5) for predicting single-cell CpG methylation states using DNA sequences. The model was applied to mouse and human cells and showed an accuracy of $AUC = 0.83$ (AUC for Area Under the Curve). Also, using DeepCpG, they were able to detect known and new motifs that are associated with methylation changes between cells. Two other methods also based on deep learning, Deepsea [ZT15] and DanQ [QX16], were published to predict the effects of non-coding variants using only DNA information. Their models are based on the same data but with different model architectures. Deepsea is based only on convolution networks, and they predict non-coding functions including histone modifications with an $AUC = 0.896$. DanQ was rather an amelioration of the Deepsea method using not only a convolution network but also recurrent network [SP97] and predicted functions using DNA sequences with $AUC = 0.927$. As well as for DeepCpG, Deepsea and DanQ capture known and new regulatory motifs.

All of these studies and many more provide evidence that DNA sequences contain information able to shape epigenome and thus gene expression.

1.4.3 Gene expression prediction using DNA sequence

In their work presented in Subsection 1.4.2, Quant and Bird proposed that proteins able to *read* domains of relatively uniform DNA base composition may modulate the epigenome and ultimately gene expression [QB16]. In line with this statement, we propose in this thesis a model to predict gene expression directly from DNA sequences.

This problem has already been tackled in yeast. Most of the studies concentrated on the relationship between gene expression and amino acid (codon) and were tested in *Saccharomyces cerevisiae*. At first, the relation between gene expression and codon was based on numerical indices. Codon adaptive index (CAI, Sharp 1987 [SL87]) and codon usage (CU, Karlin 1998 [KMC98]) are the most known numerical indicators based on frequencies of amino acids of each gene. One of the flaws of these methods is that they are applied to small sets of highly expressed genes. In 2003 a revision of these two methods was presented

by Jansen *& al.* [JBG03]. They used genome-wide yeast expression data and introduced a procedure to improve indicators parameters. An improvement of 10% of correlation was observed with the revisited method. In 2000, Coghlan and Wolfe [CW00] studied the relationship between the frequency of preferred codons in a gene and mRNA concentration. They compared different published methods (one of which is CAI) and three different mRNA concentration data sets from whole-genome studies. Using CAI they found strong correlation on different data sets (correlation(CAI) = 0.62). Last, Raghava *& al.* [RH05] used Support vector machine (SVM) based method [CV95] and predicted gene expression using amino acid and dipeptide compositions with 71% of correlation.

In the same context, in 2004 Beer and Tavazoie (BT) [BT04] attempt to predict gene expression from DNA sequence in *Saccharomyces cerevisiae*. First, they used a clustering algorithm to find 49 different sets of co-expressed genes. For each set, they identified the common DNA sequence features (motifs) among gene promoters and calculated a score for each motif (using PWM). A Bayesian Network [FK03] was used to predict gene expression in each set using calculated motif scores as well as their position and orientation. The model was applied using 5-folds cross-validation (CV) and had an accuracy of 73%. This paper was severely criticized and reexamined in 2007 by Yuan *& al.* [YGSL07]. First, Yuan *& al.* declared that the accuracy of the BT model was overestimated by 10% due to a flawed cross-validation process that does not include the step of motif identification. Besides, Yuan *& al.* criticized the choice of model that is a black box and lead to overfitting. For these reasons, Yuan *& al.* proposed to adjust the cross-validation process and to use a naive Bayes classifier [R+01] to estimate co-expressed gene expressing using motifs (same data sets from BT). They reached an accuracy of 75% with the adjusted model and CV. They also showed that both the orientation and the position of motifs do not affect the results of the naive Bayes classifier.

These studies suggest that there is a direct relationship between gene expression and DNA sequence in yeast and motivate our work aimed at predicting gene expression using DNA sequence in human using mRNA expressions from cancer patients.

1.4.4 Interactions framework in bio-statistics

Gene regulation is a complex process that requires combinations of many different biological elements such as TFs, proteins, chromatin structure, and other genomic factors. To understand gene regulation, we do not restrict the study to a single regulatory element and single gene. Instead, scientists started to explore complex relationships between different elements.

In statistics as well as in biology, interactions have attracted the attention of most statisticians. Basic models in regression do not always detect interactions, which opened a new concept of models developed specially to detect interactions between predictive variables. Different types of biological interactions (TFs, ...) were well studied in the biostatistics field. We have discussed above the importance of combinations of TFs to control gene expression. One challenge is to identify these combinations. Das [DBZ04] used the MARS model to create a model that studies the significance of both single motifs and pairs of motifs on gene expression predictions in the yeast cell. The method provides a variable selection framework via a method that fits a model by using the stepwise forward addition of linear splines (a function of the motif) and their product. Likewise, Yu *& al.* (2006) [YLM+06] identified significant interactions of pairs of TFs in *Saccharomyces cerevisiae*. The method is based on the co-occurrence of their binding sites and the distance between them on the promoter. These two papers are restrained to pairs of interactions. On the other hand, Terada *& al.* (2013) [TOHTS13] presented a statistical model based on the Fisher test as well as multiple test corrections to detect significant combinations of different motifs (2 or more). Tested on human breast cancer gene expression and using known motifs from available ChIP-seq data, they detected 23 significant motif combinations with one of an eight-motifs combination. From a statistical point of view, many methods are available to detect interactions. In the same concept, Basu & al. (2017) [BKBY18] presented the iterative Random Forests method to detect high order proteins interactions. The method is based on iterating a number of the forests with weighted features and applying on the last one decision rules on significant combinations of variables using Random Intersection Trees [SM14]. Selected interactions are stable. This method shows very high performances

in classification problems.

Sequence interactions are another type of interactions that control gene expression. Exploring the 3-dimensional chromatin as well as the DNA structure, interactions between close and distal regulatory regions (*e.g.* promoter/enhancer) is of crucial importance because they play an essential role in gene transcriptions (see [BF15] for review).

Over the years, new advanced bioinformatics and biostatistics tools were developed with the capacity of detecting sequence interactions. One of the most common sequence interactions studied using these tools is that between enhancer and promoters. In 2015, Roy *& al* [RSC+15] presented RIPPLE for Regulatory Interaction Prediction for Promoters and Long-range enhancers. The method uses machine learning to predict enhancer-promoter interactions using 5C [SLJD12] interactions on few genomic data. In the same concept, Singh [SYPM16] in 2016 and Mao [MKC17] in 2017 developed deep network (Convolution) frameworks to show that only DNA sequence features can predict long-range enhancer-promoter interactions. Similar results were found by Nikumbh [NP17] based on an SVM predictor [BGV92] and including all the chromatin sequences.

Finally, our method aims at detecting interactions between two regulatory regions (inter) or within one regulatory region (intra) on the DNA sequence. The statistical tests were inspired by the work of Wein-Yin Loh (2002) [Loh02]. He presented the GUIDE method that can select interactions of two variables as a decision rule in a tree nodes. The model is based on a χ^2 test that selects an interaction of pairs of variables if it is more significant than each variable alone. This method increased model performances by 20% compared to CART model.

Chapter 2

Accurately predicting gene expression from nucleotide composition using linear regression

We presented previously different biological elements that control gene expression at different levels of the regulation process. Besides, we presented the deregulation of these controls in cancer cells (Section 1.1.3). In this chapter, we present a framework with the aim of predicting and explaining gene expression from DNA sequences in tumors from different cancer types. In the previous chapter (Section 1.4), we presented different approaches that were developed to predict gene expression from experimental data (Section 1.4.1). Also, others approaches used the DNA sequences with the aim of predicting experimental data, more precisely epigenetics (Section 1.4.2). Our study was partially motivated by these frameworks to establish a relationship between gene expression and variables computed directly on the DNA sequence of different regulatory regions. Also, concurrent works were very recently published based on deep learning with the aim of predicting gene expression from the sequence ([ZTY+18], [AS18], [KRB+18]). Since they are very recent, these papers are not presented in the State-of-the-art but will be discussed later in the Discussion and perspectives. This project was a teamwork work with other PhD students and researchers. In this chapter, I present my contribution to this project as well as some results in Section 2.1. The article is included in Section 2.2.

2.1 Contributions and first results

The main objective of this work is to develop a model that predicts gene expression in function of variables computed on the DNA sequence. First, we chose the predictive model. For this chapter, we used a Lasso linear regression model [Tib96] (see Section 1.2.1.2 for more information). The choice of the model was based on its ability of variable selection and its simplicity of evaluation the contribution of each variable.

A part of this project was conducted by Chloé Bessière (PhD student) to evaluate different regulatory elements and regions using the Lasso penalized linear model. The choice of this model was based on a model comparison that I performed (see the following paragraph). At first, The study was restrained to gene promoters using as predictive variables scores of motifs and nucleotide compositions. The results of the different comparisons showed that nucleotide compositions presented good accuracy in predicting gene expression and outperformed the contributions of motifs that increased slightly model performances. The results of this model (based on both nucleotide compositions and motifs) were compared to those of models based on experimental data (*i.e.* RACER [LLZ14a] and TEPIC [SGG+16]) based on ChIP-seq and DNA accessibility data. Our model presented similar performances. Finally, Chloé showed the contribution of different regulatory regions (introns, CDS, ...) in predicting gene expression using a forward procedure. One of the most revealing conclusions was the high importance of introns in regulating gene expression. The model with the highest performances was a Lasso penalized linear regression fitted to predict gene expression based on nucleotide compositions in 8 different regulatory regions (160 variables in total). Based on that model, I went further in the study to explain the results and select significant variables.

Different models comparison

First, I compared the Lasso penalized regression to two other non-parametric models, Regression Trees [BFOS84] and Random Forests [Bre01] (see Sections 1.2.2.1 and 1.2.2.2). Regression Trees presented the lowest performances while Random Forests showed similar

results as the Lasso penalized linear regression. However, the Lasso penalized linear regression was favored given its low time of execution and its simplicity. Results are presented in Supplementary Table 1.

Stability selection

To go further in the study, I applied a stability selection approach to the Lasso penalized regression. This approach selected the nucleotide composition in different regions that were stable and important for predicting gene expression. The study showed that some nucleotide compositions were important despite the cancer type while others were cancer type specific. Besides, I ran a regression model using only stable variables and based on the sign of the regression coefficients of each stable variable, and I classified variables in function of their role in the regulation: activator if positive and inhibitor if negative. Despite some specificities, but our model was not cell-specific. A model fitted to predict gene expression on one tumor from a cancer type, showed similar accuracy on another tumor from a different cancer type. This concept is further developed in the Discussion and perspective Chapter.

Gene classification

Based on the residuals of the Lasso penalized linear model as the response variable and the nucleotide compositions as predictive variables, I fitted a Regression Tree, for each tumor, to classify genes based on their prediction accuracy. A set of groups of genes was poorly predicted in all cancer types. However, the other groups of genes were well predicted with low residuals either in all cancer types or a specific type. For each well-predicted group of genes, I further studied its functional annotations enrichment and succeeded to identify specific functions in each group. Also, Chloé tested the ubiquitous of the genes using the Gini coefficients.

Link to topological chromatin domains

Further validation of the model was relating the different groups found in the previous section to the DNA architecture groups, more precisely TADs (see Section 1.3) which biologically create groups of genes. Using the groups of genes classified as well predicted in all cancer types, I ran an enrichment test in each group of genes for each TADs. From this method, I showed that genes grouped in the same TADs were based on the same nucleotide compositions.

Mutations

Finally, I used the results of mutation counts presented by Lawrence *& al.* [LSP+13] to test if the stable selected variables by our model in a cancer type, were those that were highly mutated in the same type. For that, I used a hypergeometric enrichment test. This work is not presented nor published provide the high bias I detected in the results. One of the highest mutated di-nucleotides were CpG in different regulatory regions in most cancer types. In the same way, CpG in different regions were stable in almost all cancers which biased the enrichment tests. Further work is required to overcome this bias (additional information in the discussion Chapter 5)

Conclusion

In this article, we provide a framework to predict gene expression from DNA sequences using a Lasso penalized linear regression. The model shows the importance of the DNA sequence in gene regulation compared to experimental data. Furthermore, we highlight the importance of some regulatory regions, especially introns, in gene regulation. Finally, we found that the model well predicts housekeeping genes with general functions as well as some cancer-specific genes with specific biological functions.

2.2 The integral article

The article is published in PLOS Computional Biologie ,January 2, 2018

PLOS COMPUTATIONAL BIOLOGY

RESEARCH ARTICLE

Probing instructions for expression regulation in gene nucleotide compositions

Chloé Bessière[1,2☯], May Taha[1,2,3☯], Florent Petitprez[1,2], Jimmy Vandel[1,4], Jean-Michel Marin[1,3], Laurent Bréhélin[1,4‡*], Sophie Lèbre[1,3,5‡*], Charles-Henri Lecellier[1,2‡*]

1 IBC, Univ. Montpellier, CNRS, Montpellier, France, **2** Institut de Génétique Moléculaire de Montpellier, University of Montpellier, CNRS, Montpellier, France, **3** IMAG, Univ. Montpellier, CNRS, Montpellier, France, **4** LIRMM, Univ. Montpellier, CNRS, Montpellier, France, **5** Univ. Paul-Valéry-Montpellier 3, Montpellier, France

☯ These authors contributed equally to this work.
‡ LB, SL, and CHL also contributed equally to this work.
* brehelin@lirmm.fr (LB); sophie.lebre@umontpellier.fr (SL); charles.lecellier@igmm.cnrs.fr (CHL)

Citation: Bessière C, Taha M, Petitprez F, Vandel J, Marin J-M, Bréhélin L, et al. (2018) Probing instructions for expression regulation in gene nucleotide compositions. PLoS Comput Biol 14(1): e1005921. https://doi.org/10.1371/journal.pcbi.1005921

Editor: Zhaolei Zhang, University of Toronto, CANADA

Received: July 11, 2017

Accepted: December 10, 2017

Published: January 2, 2018

Data Availability Statement: All relevant data are within the paper, its Supporting Information files, and at http://www.univ-montp3.fr/miap/~lebre/IBCRegulatoryGenomics.

Funding: The work was supported by funding from CNRS, Plan d'Investissement d'Avenir #ANR-11-BINF-0002 Institut de Biologie Computationnelle (young investigator grant to CHL and post-doctoral fellowship to JV), Labex NUMEV (post-doctoral fellowship to JV), INSERM-ITMO Cancer project "LIONS" BIO2015-04. MT is a recipient of a CBS2-

Abstract

Gene expression is orchestrated by distinct regulatory regions to ensure a wide variety of cell types and functions. A challenge is to identify which regulatory regions are active, what are their associated features and how they work together in each cell type. Several approaches have tackled this problem by modeling gene expression based on epigenetic marks, with the ultimate goal of identifying driving regions and associated genomic variations that are clinically relevant in particular in precision medicine. However, these models rely on experimental data, which are limited to specific samples (even often to cell lines) and cannot be generated for all regulators and all patients. In addition, we show here that, although these approaches are accurate in predicting gene expression, inference of TF combinations from this type of models is not straightforward. Furthermore these methods are not designed to capture regulation instructions present at the sequence level, before the binding of regulators or the opening of the chromatin. Here, we probe sequence-level instructions for gene expression and develop a method to explain mRNA levels based solely on nucleotide features. Our method positions nucleotide composition as a critical component of gene expression. Moreover, our approach, able to rank regulatory regions according to their contribution, unveils a strong influence of the gene body sequence, in particular introns. We further provide evidence that the contribution of nucleotide content can be linked to co-regulations associated with genome 3D architecture and to associations of genes within topologically associated domains.

Author summary

Identifying a maximum of DNA determinants implicated in gene regulation will accelerate genetic analyses and precision medicine approaches by identifying key gene features. In that context decoding the sequence-level instructions for gene regulation is of prime importance. Among global efforts to achieve this objective, we propose a novel approach

I2S joint doctoral fellowship. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Competing interests: The authors have declared that no competing interests exist.

able to explain gene expression in each patient sample using only DNA features. Our approach, which is as accurate as methods based on epigenetics data, reveals a strong influence of the nucleotide content of gene body sequences, in particular introns. In contrast to canonical regulations mediated by specific DNA motifs, our model unveils a contribution of global nucleotide content notably in co-regulations associated with genome 3D architecture and to associations of genes within topologically associated domains. Overall our study confirms and takes advantage of the existence of sequence-level instructions for gene expression, which lie in genomic regions largely underestimated in regulatory genomics but which appear to be linked to chromatin architecture.

Introduction

The diversity of cell types and cellular functions is defined by specific patterns of gene expression. The regulation of gene expression involves a plethora of DNA/RNA-binding proteins that bind specific motifs present in various DNA/RNA regulatory regions. At the DNA level, transcription factors (TFs) typically bind 6-8bp-long motifs present in promoter regions, which are close to transcription start site (TSS). TFs can also bind enhancer regions, which are distal to TSSs and often interspersed along considerable physical distance through the genome [1]. The current view is that DNA looping mediated by specific proteins and RNAs places enhancers in close proximity with target gene promoters (for review [2–5]). High-resolution chromatin conformation capture (Hi-C) technology identified contiguous genomic regions with high contact frequencies, referred to as topologically associated domains (TADs) [6]. Within a TAD, enhancers can work with many promoters and, on the other hand, promoters can contact more than one enhancer [5, 7]. Several large-scale data derived from high-throughput experiments (such as ChIP-seq [8], SELEX-seq [9], RNAcompete [10]) can be used to highlight TF/RBP binding preferences and build Position Weight Matrixes (PWMs) [11]. The human genome is thought to encode $\sim$2,000 TFs [12] and $>$1,500 RBPs [13]. It follows that gene regulation is achieved primarily by allowing the proper combination to occur i.e. enabling cell- and/or function-specific regulators (TFs or RBPs) to bind the proper sequences in the appropriate regulatory regions. In that context, epigenetics clearly plays a central role as it influences the binding of the regulators and ultimately gene expression [14]. Provided the variety of regulatory mechanisms, deciphering their combination requires mathematical/computational methods able to consider all possible combinations [15]. Several methods have recently been proposed to tackle this problem [16–19]. Although these models appear very efficient in predicting gene expression and identifying key regulators, they mostly rely on experimental data (ChIP-seq, methylation, DNase hypersensitivity), which are limited to specific samples (often to cell lines) and which cannot be generated for all TFs/RBPs and all cell types. These technological features impede from using this type of approaches in a clinical context in particular in precision medicine. In addition, we show here that, although these approaches are accurate, their biological interpretation can be misleading. Finally these methods are not designed to capture regulation instructions that may lie at the sequence-level before the binding of regulators or the opening of the chromatin. There is indeed a growing body of evidence suggesting that the DNA sequence *per se* contains information able to shape the epigenome and explain gene expression [20–25]. Several studies have shown that sequence variations affect histone modifications [21–23]. Specific DNA motifs can be associated with specific epigenetic marks and the presence of these motifs can predict the epigenome in a given cell type [24]. Quante and Bird proposed that proteins able to "read" domains of

relatively uniform DNA base composition may modulate the epigenome and ultimately gene expression [20]. In that view, modeling gene expression using only DNA sequences and a set of predefined DNA/RNA features (without considering experimental data others than expression data) would be feasible. In line with this proposal, Raghava and Han developed a Support Vector Machine (SVM)-based method to predict gene expression from amino acid and dipeptide composition in *Saccharomyces cerevisiae* [26].

Here, we built a global regression model per sample to explain the expression of the different genes using their nucleotide compositions as predictive variables. The idea beyond our approach is that the selected variables (defining the model) are specific to each sample. Hence the expression of a given gene may be predicted by different variables in different samples. This approach was tested on several independent datasets: 2,053 samples from The Cancer Genome Atlas (1,512 RNA-sequencing data and 582 microarrays) and 3 ENCODE cell lines (RNA sequencing). When restricted to DNA features of promoter regions our model showed accuracy similar to that of two independent methods based on experimental data [17, 19]. We confirmed the importance of nucleotide composition in predicting gene expression. Moreover the performance of our approach increases by combining the contribution of different types of regulatory regions. We thus showed that the gene body (introns, CDS and UTRs), as opposed to sequences located upstream (promoter) or downstream, had the most significant contribution in our model. We further provided evidence that the contribution of nucleotide composition in predicting gene expression is linked to co-regulations associated with genome architecture and TADs.

Materials and methods

Datasets, sequences and online resources

RNA-seq V2 level 3 processed data were downloaded from the TCGA Data Portal. Our training data set contained 241 samples randomly chosen from 12 different cancers (20 cancerous samples for each cancer except 21 for LAML). Our model was further evaluated on an additional set of 1,270 tumors from 14 cancer types. We also tested our model on 582 TCGA microarray data. The TCGA barcodes of the samples used in our study have been made available at http://www.univ-montp3.fr/miap/~lebre/IBCRegulatoryGenomics.

Isoform expression data (.rsem.isoforms.normalized_results files) were downloaded from the Broad TCGA GDAC (http://gdac.broadinstitute.org) using firehose_get. We collected data for 73599 isoforms in 225 samples of the 241 initially considered. All the genes and isoforms not detected (no read) in any of the considered samples were removed from the analyses. Expression data were log transformed.

All sequences were mapped to the hg38 human genome and the UCSC liftover tool was used when necessary. Gene TSS positions were extracted from GENCODEv24. UTR and CDS coordinates were extracted from ENSEMBL Biomart. To assign only one 5UTR sequence to one gene, we merged all annotated 5UTRs associated with the gene of interest using Bedtools merge [27] and further concatenated all sequences. The same procedure was used for 3UTRs and CDSs. Intron sequences are GENCODEv24 genes to which 5UTR, 3UTR and CDS sequences described above were substracted using Bedtools substract [27]. These sequences therefore corresponded to constitutive introns. The intron sequences were concatenated per gene. The downstream flanking region (DFR) was defined as the region spanning 1kb after GENCODE v24 gene end. Fasta files were generated using UCSC Table Browser or Bedtools getfasta [27].

TCGA isoform TSSs were retrieved from https://webshare.bioinf.unc.edu/public/mRNAseq_TCGA/unc_hg19.bed and converted into hg38 coordinates with UCSC liftover.

For other regulatory regions associated to transcript isoforms (UTRs, CDS, introns and DFR), we used GENCODE v24 annotations.

Nucleotide composition

The nucleotide (n = 4) and dinucleotide (n = 16) percentages were computed from the different regulatory sequences where:

$$percentage(N, s) = \frac{\sharp N}{l}$$

is the percentage of nucleotide N in the regulatory sequence s, with N in {A, C, G, T} and l the length of sequence s, and

$$percentage(NpM, s) = \frac{\sharp NpM}{l - 1}$$

is the NpM dinucleotide percentage in the regulatory sequence s, with N and M in {A, C, G, T} and l the length of sequence s.

Motif scores

Motif scores in core promoters were computed using the method explained in [11] and Position Weight Matrix (PWM) available in JASPAR CORE 2016 database [28]. Let w be a motif and s a nucleic acid sequence. For all nucleotide N in {A, C, G, T}, we denoted by $P(N|w_j)$ the probability of nucleotide N in position j of motif w obtained from the PWM, and by $P(N)$ the prior probability of nucleotide N in all sequences.

The score of motif w at position i of sequence s is computed as follows:

$$score(w, s, i) = \sum_{j=0}^{|w|-1} \log \frac{P(s_{i+j}|w_j)}{P(s_{i+j})}$$

with $|w|$ the length of motif w, s_{i+j} the nucleotide at position $i + j$ in sequence s, The score of motif w for sequence s is computed as the maximal score that can be achieved at any position of s, i.e.:

$$score(w, s) = \max_{i=0}^{l-|w|} score(w, s, i),$$

with l the length of sequence s.

Models were also built on sum scores as:

$$scoreSum(w, s) = \sum_{i=0}^{l-|w|} score(w, s, i),$$

and further compared to models built on mean scores (S1 Fig). Taking mean or sum scores per region yielded similar results (Wilcoxon test p-value = 0.68).

DNAshape scores

DNA shape scores were computed using DNAshapeR [29]. Briefly, provided nucleotide sequences, DNAshapeR uses a sliding pentamer window to derive the structural features corresponding to minor groove width (MGW), helix twist (HelT), propeller twist (ProT) and Roll from all-atom Monte Carlo simulations [29]. Thus, for each DNA shape, a score is given to

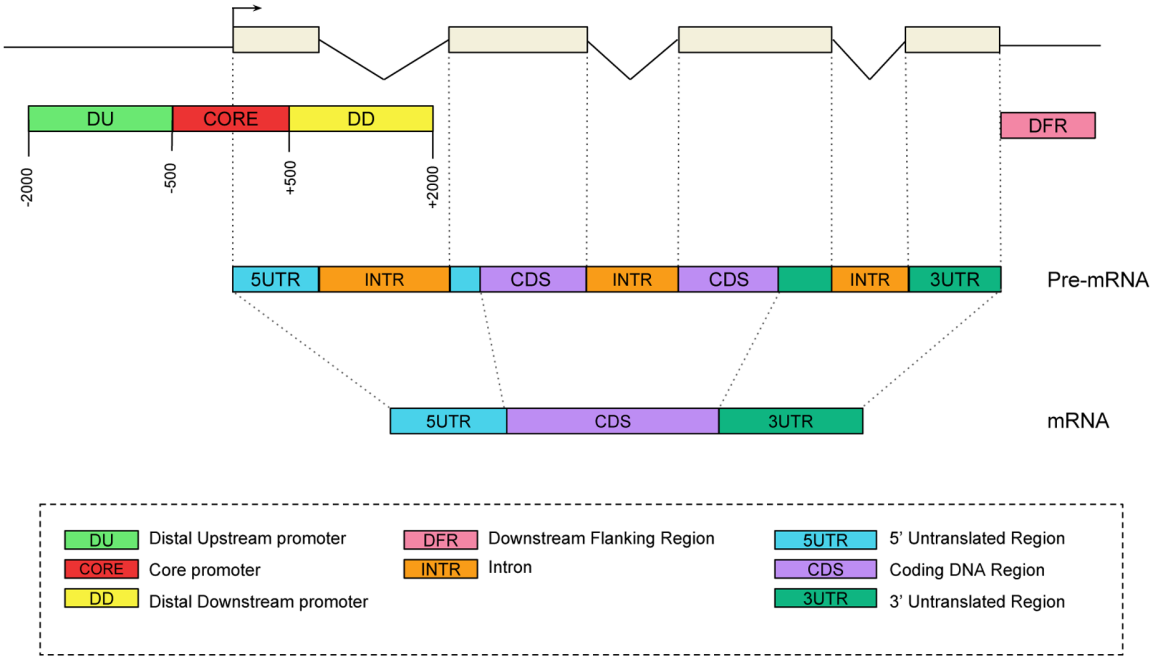


Fig 1. Genomic regions considered for gene expression prediction. An illustrative transcript is shown as example.

https://doi.org/10.1371/journal.pcbi.1005921.g001

each base of each sequence considered (DU, CORE and DD—see Fig 1). We then computed the mean of these scores for each sequence providing 12 additional variables per gene.

Enhancers

The coordinates of the enhancers mapped by FANTOM on the hg19 assembly [7] were converted into hg38 using UCSC liftover and further intersected with the different regulatory regions. We computed the density of enhancers per regulatory region (R) by dividing the sum, for all genes, of the intersection length of enhancers with gene $i(L_{enh_i})$ by the sum of the lengths of this regulatory region for all genes:

$$enhDensity_{(R)} = \frac{\sum_i (L_{enh_i} \text{ in } R_i)}{\sum_i length(R_i)}$$

Copy Number Variation (CNV)

Processed data were downloaded from the firehose Broad GDAC (https://gdac.broadinstitute.org/). We used the genome-wide SNP array data and the segment mean scores. In order to assign a CNV score to each gene, the coordinates (hg19) of the probes were intersected with that of GENCODE v19 genes using Bedtools intersect [27] and an overlap of 85% of the gene total length. The corresponding segment mean value was then assigned to the intersecting genes. In case no intersection was detected, the gene was assigned a score of 0. We next computed Spearman correlations between genes absolute error (lasso model) and genes absolute segment mean score for each of the 241 samples of the training set.

Expression quantitative trait loci and single nucleotide polymorphisms

The v6p GTex *cis*-eQTLs were downloaded from the GTex Portal (http://www.gtexportal.org/home/). The hg19 *cis*-eQTL coordinates were converted into hg38 using UCSC liftover and further intersected with the different regulatory regions. We restricted our analyses to *cis*-eQTLs impacting their own host gene. We computed the density of *cis*-eQTL per regulatory region (R) by dividing the sum, for all genes, of the number of *cis*-eQTLs of gene i ($eQTLs_i$) located in the considered region for gene i (R_i) by the sum of the lengths of this regulatory region for all genes:

$$eQTLdensity_{(R)} = \frac{\sum_i \#(eQTLs_i \text{ in } R_i)}{\sum_i length(R_i)}$$

Likewise we computed the density of SNPs in core promoters and introns by intersecting coordinates of these two regions (liftovered to hg19) with that of SNPs detected on chromosomes 1, 2 and 19 (ftp://ftp.ncbi.nih.gov/snp/organisms/human_9606_b150_GRCh37p13/BED/):

$$SNPdensity_{(R)} = \frac{\sum_i \#(SNP_i \text{ in } R_i)}{\sum_i length(R_i)}$$

Methylation

Illumina Infinium Human DNA Methylation 450 level 3 data were downloaded from the Broad TCGA GDAC (http://gdac.broadinstitute.org) using firehose_get. The coordinates of the methylation sites (hg18) were converted into hg38 using the UCSC liftover and further intersected with that of the core promoters (hg38). For each gene, we computed the median of the beta values of the methylation sites present in the core promoter and further calculated the median of these values in 21 LAML and 17 READ samples with both RNA-seq and methylation data. We compared the overall methylation status of the core promoters in LAML and READ using a wilcoxon test.

Gini coefficient

We used 8,556 GTEx RNA-seq libraries (https://www.gtexportal.org/home/datasets) to compute the Gini coefficient for 16,134 genes on the 16,294 considered in our model. Gini coefficient measures statistical dispersion and can be used to measure gene ubiquity: value 0 represents genes expressed in all sam- ples while value 1 represents genes expressed in only one sample. To compute Gini coefficient we used `R` package `ineq`. We then computed, for the 241 samples, Spearman correlation between Gini coefficients and model gene absolute errors. Similar analyses were performed with 1,897 FANTOM 5 CAGE libraries to compute the Gini coefficients for 15,904 genes.

Functional enrichment

Gene functional enrichments were computed using the database for annotation, visualization and integrated discovery (DAVID) [30].

Linear regression with ℓ_1-norm penalty (Lasso)

We performed estimation of the linear regression model (1) via the lasso [31]. Given a linear regression with standardized predictors and centered response values, the lasso solves the

ℓ_1-penalized regression problem of finding the vector coefficient $\beta = \{\beta_i\}$ in order to minimize

$$Min\left(||y^c(g) - \sum_i \beta_i x^s_{i,g}||^2 + \lambda \sum_i |\beta_i|\right),$$

where $y^c(g)$ is the centered gene expression for all gene g, $x^s_{i,g}$ is the standardized DNA feature i for gene g and $\sum_i |\beta_i|$ is the ℓ_1-norm of the vector coefficient β. Parameter λ is the tuning parameter chosen by 10 fold cross validation. The higher the value of λ, the fewer the variables. This is equivalent to minimizing the sum of squares with a constraint of the form $\sum_i |\beta_i| \leq s$. Gene expression predictions are computed using coefficient β estimated with the value of λ that minimizes the mean square error. Lasso inference was performed using the function `cv.glmnet` from the `R` package `glmnet` [32]. The LASSO model was compared to two non parametric approaches: Regression trees (CART) [33] and Random forest [34]. S1 Table summarizes accuracy and computing time of each approach. Regression trees achieved significantly lower accuracy than the two other approaches (Wilcox test p-values $< 2e^{-16}$), while linear model and random forest yielded similar results (p-value 0.18). Moreover, computing time for linear model was much lower than that of random forest. These results emphasize the merits of linear model such as LASSO in their interpretability and efficiency.

Variable stability selection

We used the stability selection method developed by Meinshausen *et al.* [35], which is a classical selection method combined with lasso penalization. Consistently selected variables were identified as follows for each sample. First, the lasso inference is repeated 500 times where, for each iteration, (i) only 50% of the genes is used (uniformly sampled) and (ii) a random weight (uniformly sampled in [0.5;1]) is attributed to each predictive variable. Second, a variable is considered as stable if selected in more than 70% of the iterations, using the method proposed in [36] to set the value of lasso penalty λ. One of the advantage of this method is that the variable selection frequency is computed globally for all the variables by attributing a random weight to each variable at each iteration, thus taking into account the dependencies between the variables. This variable stability selection procedure was implemented using functions `stabpath` and `stabsel` from the `R` package `C060` for `glmnet` models [36].

Regression trees

Regression trees were implemented with the `rpart` package in `R` [32]. In order to avoid overfitting, trees were pruned based on a criterion chosen by cross validation to minimize mean square error. The minimum number of genes was set to 100 genes per leaf.

TAD enrichment

We considered TADs mapped in IMR90 cells [6] containing more than 10 genes (373 out of 2243 TADs with average number of genes = 14). The largest TAD had 76 associated genes. First, for each TAD and for each region considered, the percentage of each nucleotide and dinucleotide associated to the embedded genes were compared to that of all other genes using a Kolmogorov-Smirnov (KS) test. For a given dinucleotide (for example CpG), we applied KS tests to assess whether the CpG frequency distribution in genes in one specific TAD differs from the distribution in genes in other TADs. Correction for multiple tests was applied using the False Discovery Rate (FDR) < 0.05 [37] and the `R` function `p.adjust` [32]. Second, for each of the 967 groups of genes (identified by the regression trees, with mean error $<$ mean error of the 1st quartile), the over-representation of each TAD within each group was tested

using the R hypergeometric test function phyper [32]. Correction for multiple tests was applied using FDR< 0.05 [37].

Availability of data and materials

The matrices of predicted variables (log transformed RNA seq data) and predictive variables (nucleotide and dinucleotide percentages, motifs and DNA shape scores computed for all genes as described above) as well as the TCGA barcodes of the 241 samples used in our study have been made available at http://www.univ-montp3.fr/miap/~lebre/IBCRegulatoryGenomics.

Results

Mathematical approach to model gene expression

We built a global linear regression model to explain the expression of genes using DNA/RNA features associated with their regulatory regions (e.g. nucleotide composition, TF motifs, DNA shapes):

$$y(g) = a + \sum_i b_i x_{i,g} + e(g) \quad (1)$$

where $y(g)$ is the expression of gene g, $x_{i,g}$ is feature i for gene g, $e(g)$ is the residual error associated with gene g, a is the intercept and b_i is the regression coefficient associated with feature i.

The advantage of this approach is that it allows to unveil, into a single model, the most important regulatory features responsible for the observed gene expression. The relative contribution of each feature can thus be easily assessed. It is important to note that the model is specific to each sample. Hence the expression of a given gene may be predicted by different variables depending on the sample. Our computational approach was based on two steps. First, a linear regression model (1) was trained with a lasso penalty [31] to select sequence features relevant for predicting gene expression. Second, the performances of our model was evaluated by computing the mean square of the residual errors, and the correlation between the predicted and the observed expression for all genes. This was done in a 10 fold cross-validation procedure. Namely, in all experiments hereafter, the set of genes was randomly split in ten parts. Each part was alternatively used for the test (i.e. for comparing observed and predicted values) while the remaining genes were used to train the model. This ensures that the model used to predict the expression of a gene has not been trained with any information relative to this gene. Our approach was applied to a set of RNA sequencing data from TCGA. We randomly selected 241 gene expression data from 12 cancer types (see http://www.univ-montp3.fr/miap/~lebre/IBCRegulatoryGenomics for the barcode list). For each dataset (i.e sample), a regression model was learned and evaluated. See Materials and methods for a complete description of the data, the construction of the predictor variables and the inference procedure. We further evaluated our model on 3 independent ENCODE RNA-seq, 1,270 TCGA RNA-seq and 582 microarrays datasets (see below).

Contribution of the promoter nucleotide composition

We first evaluated the contribution of promoters, which are one of the most important regulatory sequences implicated in gene regulation [38]. We extracted DNA sequences encompassing ±2000 bases around all GENCODE v24 TSSs and looked at the percentage of dinucleotides along the sequences (S2 Fig). Based on these distributions, we segmented the promoter into three distinct regions: -2000/-500 (referred here to as distal upstream promoter, DU), -500/+500 (thereafter called core promoter though longer than the core promoter traditionally

considered) and +500/+2000 (distal downstream promoter, DD)(Fig 1). We computed the nucleotide (n = 4) and dinucleotide (n = 16) relative frequencies in the three distinct regions of each gene. For each sample, we trained one model using the 20 nucleotide/dinucleotide relative frequencies from each promoter segment separately, and from each combination of promoter segments. We observed that the core promoter had the strongest contribution compared to DU and DD (Fig 2A). Considering promoter as one unique sequence spanning -2000/+2000 around TSS achieved lower model accuracy than combining different promoter segments (Fig 2A). The highest accuracy was obtained combining all three promoter segments (Fig 2A).

Promoters are often centered around the 5' most upstream TSS (i.e. gene start). However genes can have multiple transcriptional start sites. The median number of alternative TSSs for the 19,393 genes listed in the TCGA RNA-seq V2 data is 5 and only 2,753 genes harbor a single TSS (S3 Fig). We therefore evaluated the performance of our model comparing different promoters centered around the first, second, third and last TSS (Fig 2B). In the absence of second TSS, we used the first TSS and likewise the second TSS in the absence of a third TSS. The last TSS represents the most downstream TSS in all cases. We found that our model achieved higher predictive accuracy with the promoters centered around the second TSS (Fig 2B), in agreement with [16]. As postulated by Cheng *et al.* [16] in the case of TFs, the nucleotide composition around the first TSS may be linked to the recruitment of chromatin remodelers and thereby prime the second TSS for gene expression. Dedicated experiments would be required to assess this point.

We noticed that incorporating the number of TSSs associated with each gene drastically increased the performance of our model (S4 Fig). Multiplying TSSs may represent a genuine mechanism to control gene expression level. On the other hand this effect may merely be due to the fact that the more a gene is expressed, the more its different isoforms will be detected (and hence more TSSs will be annotated). Because the number of known TSSs results from annotations deduced from experiments, we decided not to include this variable into our final model.

Contribution of specific features associated with promoters

Provided the importance of CpGs in promoter activity [38], we first compared our model with a model built only on promoter CpG content. We confirmed that CpG content had an important contribution in predicting gene expression (median R = 0.417, Fig 2C). However considering other dinucleotides achieved better model performances, indicating that dinucleotides other than CpG contribute to gene regulation. This is in agreement with results obtained by Nguyen *et al.*, who showed that CpG content is insufficient to encode promoter activity and that other features might be involved [39].

We integrated TF motifs considering Position Weight Matrix scores computed in the core promoter and observed a slight but significant increase of the regression performance (median r = 0.543 with motif scores vs. r = 0.502 without motif scores, Fig 2D). As DNA sequence is intrinsically linked to three-dimensional local structure of the DNA (DNA shape), we also computed, for each promoter segment (DU, CORE and DD), the mean scores of the four DNA shape features provided by DNAshapeR [29] (helix twist, minor groove width, propeller twist, and Roll), adding 12 variables to the model. Although the difference between models with and without DNA shapes is also significant, the increase in performance is more modest than when including TF motif scores (Fig 2D).

Our model suggested that nucleotide composition had a greater contribution in predicting gene expression compared to TF motifs and DNA shapes. This is in agreement with the

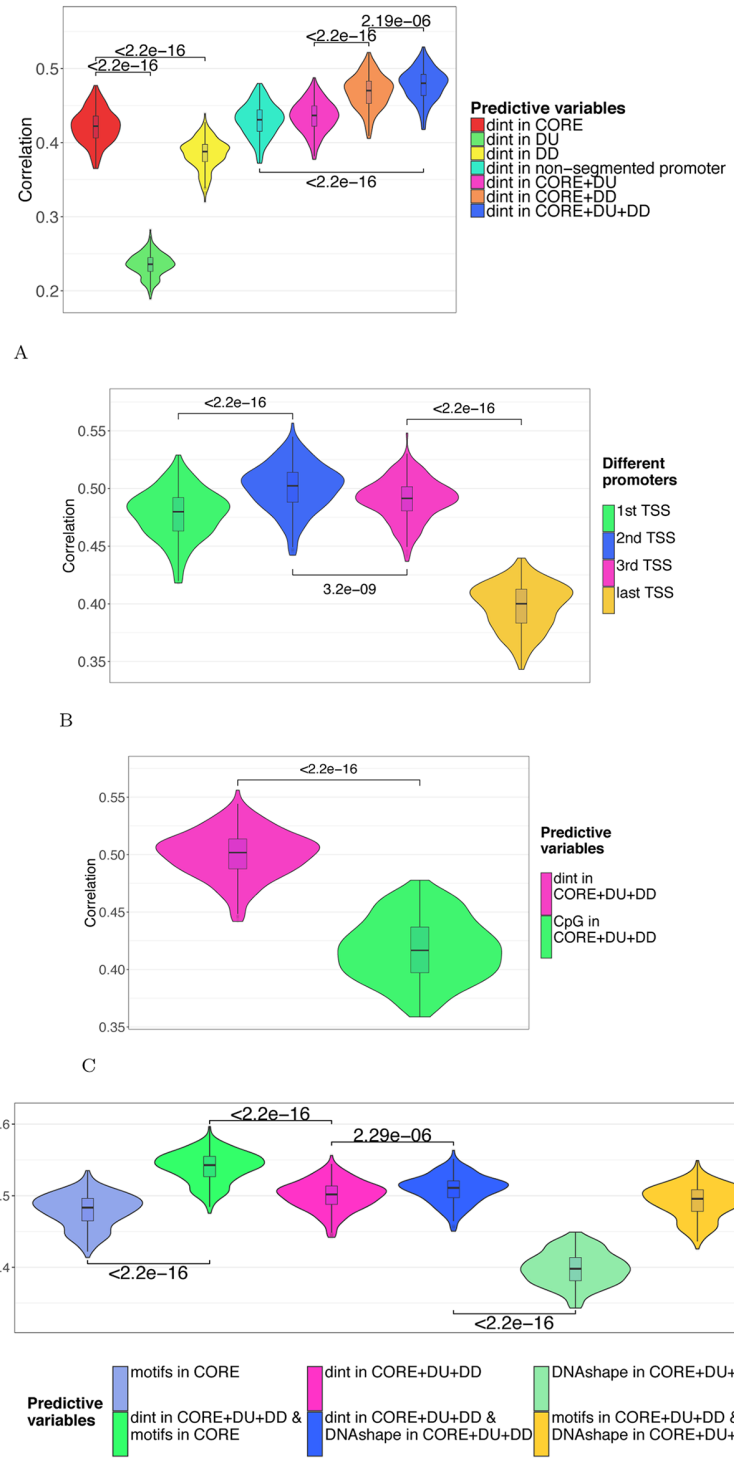


Fig 2. A: Contribution of the promoter segments. The model was built using 20 variables corresponding to the nucleotide (4) and dinucleotide (16) percentages computed in the CORE promoter (red), DU (green) or DD (yellow). These variables were then added in different combinations: CORE+DU (pink, 40 variables); CORE+DD (orange, 40 variables); CORE+DU+DD (light blue, 60 variables). Promoter segments were centered around the first most upstream TSS. For sake of comparison, the model was also built on 20 variables corresponding to the nucleotide and

dinucleotide compositions of the non segmented promoters (-2000/+2000 around the first most upstream TSS)(light blue). All different models were fitted on 19,393 genes for each of the 241 samples considered. The prediction accuracy was evaluated in each sample by evaluating the Spearman correlation coefficients between observed and predicted gene expressions in a cross-validation procedure. The correlations obtained in all samples are shown as violin plots. **B: Prediction accuracy comparing alternative TSSs.** The model was built using the 60 nucleotide/dinucleotide percentages computed in the 3 promoter segments (CORE+DU+DD) centered around 1st, 2nd, 3rd and last TSSs (from left to right). **C: Contribution of CpG.** The model was built using the 60 nucleotide/dinucleotide or only the 3 CpG percentages computed in the 3 promoter segments (CORE+DU+DD) centered around the 2nd TSS. **D: Contribution of motifs and local DNA shapes.** The model was built using (i) 60 nucleotide/dinucleotide percentages computed in the 3 promoter segments (CORE+DU+DD) ("dint", pink),(ii) 471 JASPAR2016 PWM scores computed in the CORE segment ("motifs", light blue) and (iii) the 12 DNA shapes corresponding to the 4 known DNAshapes computed in CORE, DU and DD ("DNAshape", green). All sequences were centered around the 2nd TSS. These variables were further added in different combinations to build the models indicated: dint+motifs (531 variables, green), dint+DNAshapes (32 variables, dark blue), motifs+DNAshapes (483 variables, light green).

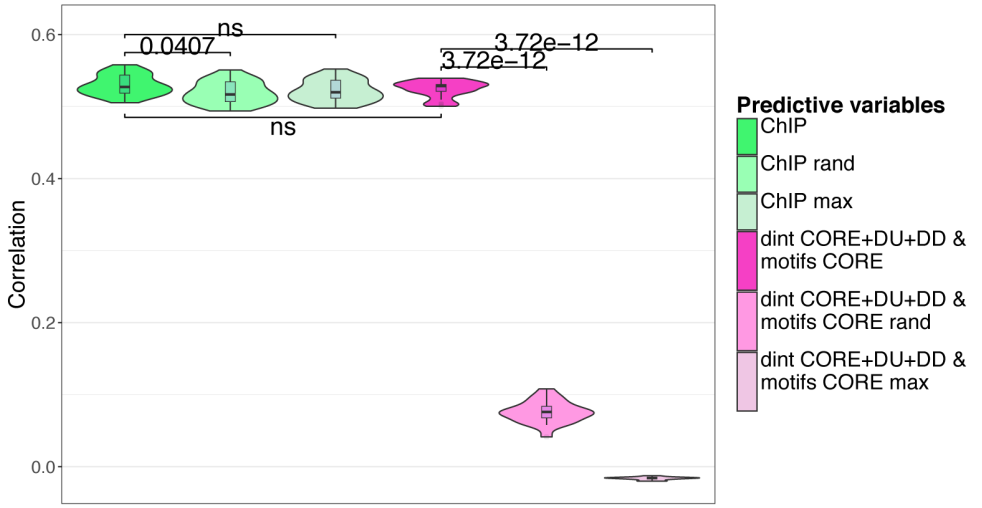
https://doi.org/10.1371/journal.pcbi.1005921.g002

findings revealing the influence of the nucleotide environment in TFBS recognition [40]. Note however that nucleotide composition, TF motifs and DNA shapes may be redundant variables. Besides, a linear model may not be optimal to efficiently capture the contributions of TF motifs and/or DNA shapes. The highest performance was achieved by combining nucleotide composition with TF motifs (Fig 2D). In the following analyses, the model was built on both dinucleotide composition and core promoter TF motifs.

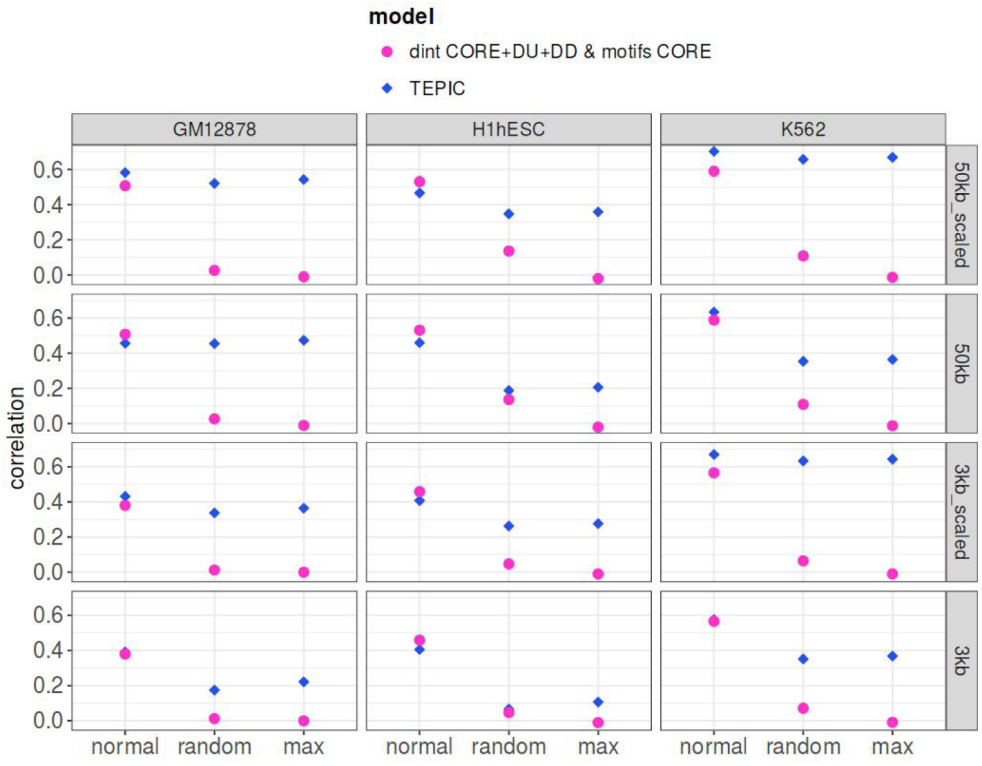
Comparison with models based on experimental data

The wealth of TF ChIP-seq, epigenetic and expression data has allowed the development of methods aimed at predicting gene expression based on differential binding of TFs and epigenetic marks [16–19]. We sought to compare our approach, which does not necessitate such cell-specific experimental data, to these methods. We first compared our results to that of Li *et al.* who used a regression approach called RACER to predict gene expression on the basis of experimental data, in particular TF ChIP-seq data and DNA methylation [17]. Note that, with this model, the contribution of TF regulation in predicting gene expression is higher than that of DNA methylation [17].

We computed the Spearman correlations between expressions observed in the subsets of LAMLs studied in [17] and expressions predicted by our model or by RACER (Fig 3A). For the sake of comparison, we used the RACER model built solely on ChIP-seq data, hereafter referred to as "ChIP-based model". RACER performance was assessed using the same cross-validation procedure we used for our method. Overall our model was as accurate as ChIP-based model (median correlation r = 0.529 with our model vs. median r = 0.527 with ChIP-based model (Fig 3A)). We then controlled the biological information retrieved by the two approaches by randomly permuting, for each gene, the values of the predictive variables (dinucleotide counts/motif scores in our model and ChIP-seq signals in the ChIP-based model). This creates a situation where the links between the combination of predictive variables and expression is broken, while preserving the score distribution of the variables associated with each gene. For example, genes associated with numerous ChIP-seq peaks will also have numerous ChIP-seq peaks in random data. In such situation, a regression model is expected to poorly perform. Surprisingly, the accuracy of ChIP-based model was not affected by the randomization process (median r = 0.517, Fig 3A) while that of our model was severely impaired (median r = 0.076, Fig 3A). We built another control model using a single predictive variable per gene corresponding to the maximum value of all predictive variables initially considered. Here again the ChIP-based model was not affected by this process (median r = 0.520, Fig 3A) while our model failed to accurately predict gene expression with this type of control variable (median r = -0.016, Fig 3A).



A



B

Fig 3. A: Comparison with model integrating TF-binding signals. The model was built using 531 variables corresponding to the 60 nucleotide/dinucleotide percentages and the 471 motif scores computed in the 3 promoter segments (CORE, DU, DD) centered around the 2nd TSS (pink). A model built on ChIP-seq data [17] was used for comparison (green). Both models were fitted on the same gene set (n = 16,298) for 21 LAML samples and assessed by cross-validation. The correlations obtained with ChIP-based RACER and our model were compared using Wilcoxon test but no significant difference was observed (p-value = 0.425). The two models were also built on randomized values of predictive variables (rand) and on the maximum value of all predictive variables (max). **B: Comparison with model integrating open-chromatin signals.** The linear model was built using the 531 variables (nucleotide/dinucleotide percentages and motif scores in CORE, DU and DD) and the expression data obtained in K562, hESC and GM12878 [19]. TEPIC was built as described in [19], within a 3 kb or a 50 kb window around TSSs. The scaled version of TEPIC

incorporates the abundance of open-chromatin peaks in the analyzed sequences. All types of TEPIC models were tested (3kb, 3kb-scaled, 50kb and 50kb-scaled) by cross-validation. In each case, our model was built on the set of genes considered by TEPIC. TEPIC uses 12 conditions making hard to compute Wilcoxon tests. A direct comparison showed that, in "normal" conditions (first column of each panel), our model and TEPIC give overall very similar results (our model being as accurate as TEPIC in 2 conditions and slightly better in 5 out of the 10 remaining conditions). Models were further built on randomized values of predictive variables (rand) and on the maximum value of all predictive variables (max). Overall, absence of effect of the randomization procedure suggests that RACER and TEPIC mainly capture the level of chromatin opening rather than the TF combinations responsible for gene expression.

https://doi.org/10.1371/journal.pcbi.1005921.g003

ChIP-seq data are probably the best way to measure the activity of a TF because binding of DNA reflects the output of RNA/protein expression as well as any appropriate post-translational modifications and subcellular localizations. However this type of data also reflects chromatin accessibility (i.e. most TFs bind accessible genomic regions) and TFs tend to form clusters on regulatory regions [41]. The binding of one TF in the promoter region is therefore likely accompanied by the binding of others. Hence, rather than inferring the TF combination responsible for gene expression, linear models based of ChIP-seq data predominantly captures the quantity of TFs (i.e. the opening of the chromatin) in the promoter region of each gene, which explains their good accuracy on randomized or maximized variables.

We indeed observed a similar bias in the results obtained by TEPIC [19], a regression method that predicts gene expression from PWM scores and open-chromatin data. Specifically, TEPIC computes a TF-affinity score for each gene and each PWM by summing up the TF affinities in all open-chromatin peaks (DNaseI-seq) within a close (3,000 bp) or large (50,000 bp) window around TSSs. This scoring takes into account the scores of PWMs in the open-chromatin peaks but is also influenced by the number of open-chromatin peaks in the analyzed sequences and the abundance of open-chromatin peaks ("scaled" version). As a result, genes with many open-chromatin peaks tend to get higher TF-affinity scores than genes with low number of open-chromatin peaks. We trained linear models on three cell-lines using either the four TEPIC affinity-scores or our variables and compared the results (Fig 3B). As for the ChIP-based models, we observed that our model was approximately as accurate as TEPIC score model, validating our approach with an independent dataset. Applying the random permutations on the TEPIC scores did not significantly impact the accuracy of the approach in most cases, especially for the scaled versions (Fig 3B). Hence, as for the ChIP-based model, the TEPIC score model seems to mainly capture the level of chromatin opening rather than the TF combinations responsible for gene expression. Conversely, our model solely built on DNA sequence features is not influenced by the chromatin accessibility and thus can yield relevant combinations of explanatory features (see the randomized control in Fig 3A and 3B). Note that the non-scaled version of TEPIC did show a loss of accuracy for cell-line H1-hESC (as well as a moderate loss for K562, but none for GM12878) when randomizing or maximizing the variables (Fig 3B). This result indicates that, although taking the abundance of open-chromatin peaks in the analyzed sequences does increase expression prediction accuracy, it might generate more irrelevant combinations of explanatory features than non-scaled versions.

Contribution of additional genomic regions

Additional genomic regions were integrated into our model. We first thought to consider enhancer sequences implicated in transcriptional regulation. We used the enhancer mapping made by the FANTOM5 project, which identified 38,554 human enhancers across 808 samples [7]. This mapping uses the CAGE technology, which captures the level of activity for both promoters and enhancers in the same samples. It is then possible to predict the potential target genes of the enhancers by correlating the activity levels of these regulatory regions over

hundreds of human samples [7]. However FANTOM5 enhancers are only assigned to 11,359 genes from the TCGA data, which correspond to the most expressed genes across different cancers (S5 Fig). Provided that the detection of enhancers relies on their activity, it is expected that enhancers are better characterized for the most frequently expressed genes. Because considering only the genes with annotated enhancers would considerably reduce the number of genes and including enhancers features only when available would introduce a strong bias in the performance of our model, we decided not to include these regulatory regions.

Second we analyzed the contribution of regions defined at the RNA level, namely 5'UTR, CDS, 3'UTR and introns, which can be responsible for post-transcriptional regulations [13, 17, 26, 42–50] (Fig 1). For all genes, we extracted all annotated 5'UTRs, 3'UTRs and CDSs, which were further merged and concatenated to a single 5'UTR, a single CDS, and a single 3'UTR per gene. Introns were defined as the remaining sequence (Fig 1). We also tested the potential contribution of the 1kb region located downstream the gene end, called thereafter Downstream Flanking Region (DFR, Fig 1). Our rationale was based on reports showing the presence of transient RNA downstream of polyadenylation sites [51], the potential presence of enhancers [7] and the existence of 5' to 3' gene looping [52].

We used a forward selection procedure by adding one region at a time: (i) all regions were tested separately and the region leading to the highest Spearman correlation between observed and predicted expression was selected as the 'first' seed region, (ii) each region not already in the model was added separately and the region yielding the best correlation was selected ('second region'), (iii) the procedure was repeated till all regions were included in the model. The correlations computed in a cross-validation procedure at each steps are indicated in S2 Table. As shown in Fig 4, the nucleotide composition of intronic sequences had the strongest contribution in the accuracy of our model, followed by UTRs (5' then 3') and CDS (Fig 4). The

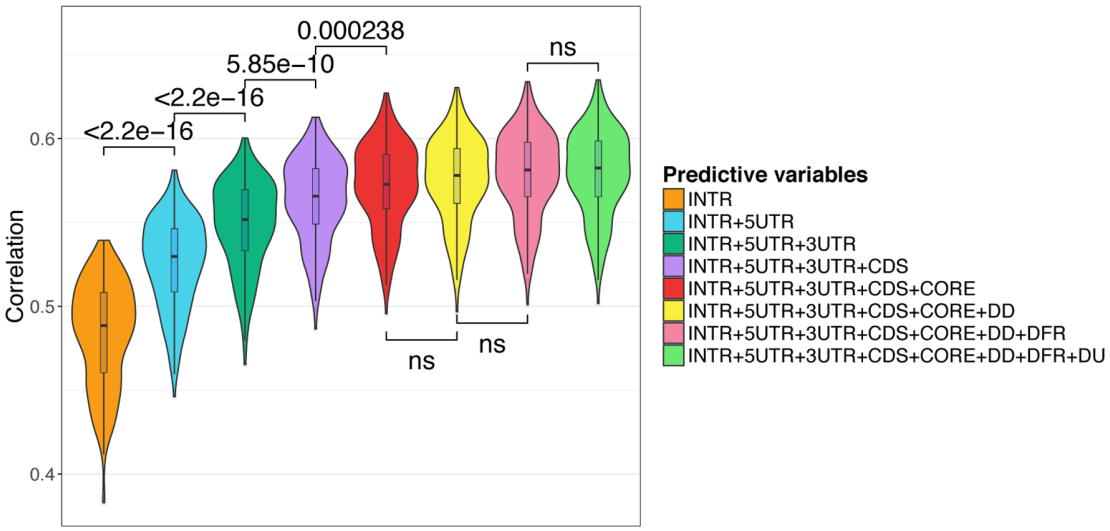


Fig 4. Contribution of additional genomic regions. Genomic regions were ranked according to their contribution in predicting gene expression. First, all regions were tested separately. Introns yielded the highest Spearman correlation between observed and predicted expressions (in a cross-validation procedure) and was selected as the 'first' seed region. Second, each region not already in the model was added separately. 5'UTR in association with introns yielded the best correlation and was therefore selected as the 'second' region. Third, the procedure was repeated till all regions were included in the model. The contribution of each region is then visualized starting from the most important (left) to the less important (right). Note that the distance between the second TSS and the first ATG is > 2000 bp for only 189 genes implying that 5'UTR and DD regions overlap. The correlations computed at each steps are indicated in (S2 Table). ns, non significant.

https://doi.org/10.1371/journal.pcbi.1005921.g004

nucleotide composition of core promoter moderately increased the prediction accuracy. In contrast the composition of regions flanking core promoter (DU and DD, Fig 1) as well as regions located downstream the end of gene (DFR, Fig 1) did not significantly improve the predictions of our model. Note that combining all regions improved the performance of our model compared to promoter alone (compare Figs 2B and 4).

We compared models built on ssDNA and dsDNA, and ssDNA-based models yielded better accuracy S6 Fig. We also compared models built on percentages of nucleotides (n = 4), dinucleotides (n = 16) and nucleotides+dinucleotides (n = 20). As shown S7A Fig, dinucleotides provided stronger prediction accuracy than nucleotides and the best accuracy was obtained combining both nucleotides and dinucleotides. We also built a model on trinucleotide percentage (n = 64) (S7A Fig). This model did yield better results than model built on nucleotide+dinucleotide. However, the correlation increase was not as important as that observed when adding dinucleotides to nucleotides. Besides, the model built on trinucleotides involves more variables and is computationally demanding. We compared models built on nucleotides+dinucleotides adding individually trinucleotide percentages of each region (i.e. 8 models built on nucleotides+dinucleotides in all regions + trinucleotides in one specific region) (S7B Fig). This analysis revealed that the correlation increase observed when incorporating trinucleotides was mostly due to the contribution of trinucleotides computed in introns, reinforcing our conclusions regarding the importance of sequence-level instructions located in this region.

Because RNA-associated regions (introns, UTRs, CDSs) had greater contribution to the prediction accuracy compared to DNA regions (promoters, DFR), we compared the accuracy of our model in predicting gene vs. transcript expression. We retrieved the normalized results for gene expression (RNAseqV2 rsem.genes.normalized_results) and the matched normalized expression signal of individual isoforms (RNAseqV2 rsem.isoforms.normalized_results) for 225 TCGA samples. Accordingly, we generated a set a predictive variables specific to each isoform (see Material and methods). We found that models built on isoforms are less accurate than models built on genes (median r = 0.35, S8 Fig and (S3 Table)). Focusing on the broad nucleotide composition may not be optimal to model isoform expression and to differentiate expression of one isoform from another. Yet another simple explanation could be that reconstructing and quantifying full-length mRNA transcripts is a difficult task, and no satisfying solution exists for now [53]. Consequently isoform as opposed to gene expression is more difficult to measure and thus to predict.

Additional validation of the model

In the above sections, our complete model, built on 160 variables corresponding to 4 nucleotide and 16 dinucleotide rates in 8 distinct regions (Fig 1), was trained with a data set containing 241 RNA-seq samples randomly chosen from 12 different cancers, and on 3 independent ENCODE RNA-seq datasets (see TEPIC comparison). We further evaluated our approach using two independent additional datasets: (a) a set of 1,270 RNA-seq samples collected from 14 cancer types and (b) a set of 582 microarray data. Overall, the RNA-seq and the microarray samples were collected from respectively 109 and 41 source sites and sequenced in 3 analysis centers. Similar accuracy was observed in all datasets (S9 and S10 Figs). Note that the correlations computed with microarray data were lower than that computed with RNA-seq data but involved lower number of genes (9,791 genes in microarrays vs. 16,294 in RNA-seq). For sake of comparison, we restricted RNA-seq data to the 9,791 microarray genes and we observed similar correlation (S10 Fig). Because our model was built on human reference genome, we also have computed the Spearman correlations between absolute values of CNV segment

mean scores and model prediction errors calculated for each gene in 241 samples corresponding to 12 cancer types. The median correlation was -0.014, arguing against the model performance being related to CNV-density (S11 Fig).

Selecting DNA features related to gene expression

We sought the main DNA features related to gene expression. The complete model built on all 8 regions (160 variables) selected $\sim$ 129 predictive variables per sample. We used the stability selection algorithm developed by Meinshausen *et al.* [35] to identify the variables that are consistently selected after data subsampling (see Materials and methods for a complete description of the procedure). This procedure selected a median of $\sim$ 16 variables per sample. The barplot in Fig 5A shows, for each variable, the proportion of samples in which the variable is selected with high consistency ($>$ 70% of the subsets).

We next determined whether stable variables exert a positive (activating) or a negative (inhibiting) effect on gene expression. For each sample, we fitted a linear regression model predicting gene expression using only the standardized variables that are stable for this sample. The activating/inhibiting effect of a variable is then indicated by the sign of its regression coefficient: <0 for a negative effect and >0 for a positive effect. The outcome of these analyses for all variables and all samples is shown Fig 5B. With the noticeable exception of CpG in the core promoter, all stable variables had an invariable positive (e.g. GpT in introns) or negative (e.g. CpA in DD and in 5UTR) contribution in gene expression prediction in all samples. In contrast, CpG in the core promoter had an alternating effect being positive in LAML and LGG for instance while negative in READ. It is also the only variable with a regression coefficient close to 0 (absolute value of median = 0.1, see S12 Fig), providing a partial explanation for the observed changes. As CpG methylation inhibits gene expression [38], we also investigated potential differences in core promoter methylation in LAML (positive contribution of CpG_CORE) and READ (negative contribution of CpG_CORE). We used the Illumina Infinium Human DNA Methylation 450 made available by TCGA and focused on the estimated methylation level (beta values) of the sites intersecting with the core promoter. We noticed that core promoters in LAML were overall more methylated (median = 0.85) than in READ (median = 0.69, wilcoxon test p-value $<$ 2.2e-16), opposite to the sign of CpG coefficient in LAML (positive contribution of CpG_CORE) and READ (negative contribution of CpG_CORE). This argued against a contribution of methylation in the alternating effect of CpG_CORE.

We observed that the accuracy of our model varied between cancer types (S9 Fig). In order to characterize well predicted genes in each sample, we used a regression tree [54] to classify genes according to the prediction accuracy of our model (i.e. absolute error). The nucleotide and dinucleotide compositions of the various considered regions were used as classifiers. This approach identified groups of genes with similar (di)nucleotide composition in the regulatory regions considered and for which our model showed similar accuracy (S13 Fig). Implicitly, it identified the variables associated with a better or a poorer prediction. We applied this approach to the 241 linear models. The number of groups built by a regression tree differs from one sample to another (average number = 14). The resulting 3,680 groups can be visualized in the heatmap depicted in Fig 6, wherein each column represents a sample and each line corresponds to a group of genes identified by a regression tree. This analysis showed that our model is not equally accurate in predicting the expression of all genes but mainly fits certain classes of genes (bottom rows of the heatmap, Fig 6) with specific genomic features (S13 Fig). Note that the groups well predicted in all cancers presumably correspond to highly and ubiquitously expressed housekeeping genes: groups with low prediction error in all samples and

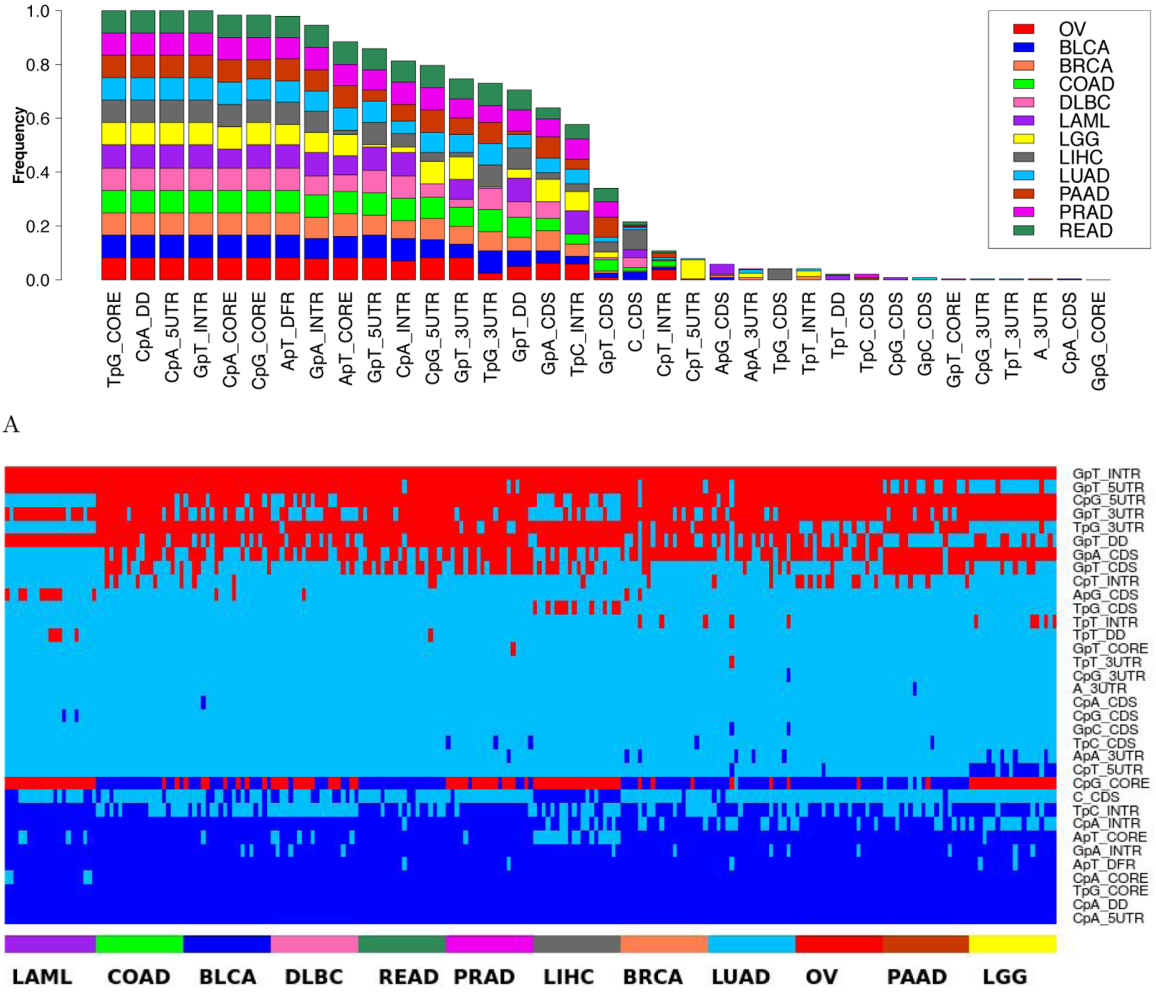


Fig 5. A: Consistently selected variables among 12 types of cancer. For each variable, the fraction of samples in which the variable is considered as stable (i. e. selected in more than 70% of the subsets after subsampling) is shown. Each color refers to a specific type of cancer. Only variables consistently selected in at least one sample are shown (out of the 160 variables). See Materials and methods for stable variable selection procedure and cancer acronyms. **B: Biological effect of the stable variables**. For each of the 241 samples (columns), a linear model was fitted using the variables (rows) stable for this sample only. The sign of the contribution of each variable in each sample is represented as follows: red for positive contribution, dark blue for negative contribution and sky blue refers to variables not selected (i.e. not stably selected for the considered sample). Only the variables stable in at least one sample are represented. Cancers and samples from the same cancer types are ranked by decreasing mean error of the linear model.

https://doi.org/10.1371/journal.pcbi.1005921.g005

cancer types (see S13 Fig for an example group of 996 genes identified by a regression tree learned in one PRAD sample) are functionally enriched for general and widespread biological processes (S4 Table). In contrast, groups well predicted in only certain cancers were associated to specific biological function. For instance, a regression tree learned on one PAAD sample identified a group of 1,531 genes, which has low prediction error in LGG and PAAD samples but high error in LAML, LIHC and DLBC samples (Fig 6 and S13 Fig). Functional annotation of this group showed that, in contrast to the group described above (S13 Fig and S4 Table), this group is also linked to specific biological processes (S5 Table).

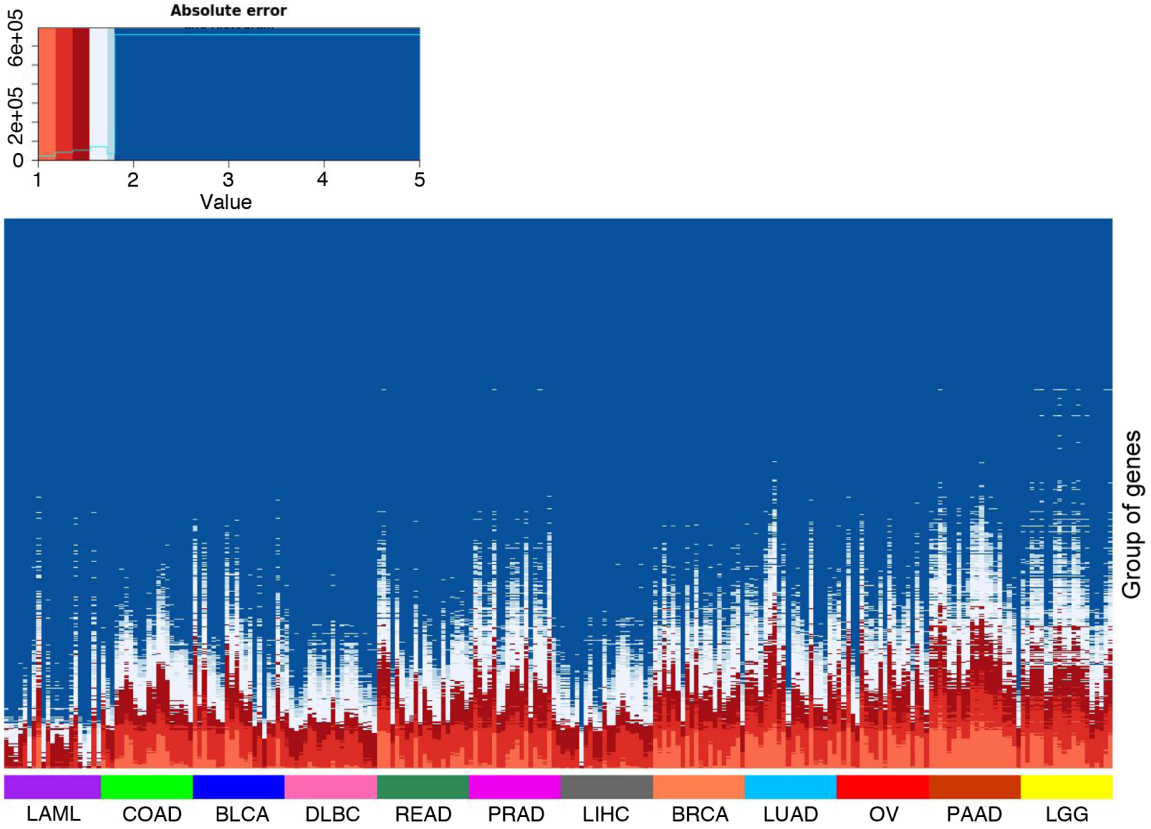


Fig 6. Gene classification according to prediction accuracy. Columns represent the various samples gathered by cancer type. Samples from the same cancer type are ranked by decreasing mean squared prediction error. Lines represent the 3,680 groups of gene obtained with the regression trees (one tree for each of the 241 samples) ranked by decreasing mean squared prediction error. Groups gathering the top 25% well predicted genes (error $< \sim 1.77$) are indicated in red and light blue.

https://doi.org/10.1371/journal.pcbi.1005921.g006

We further computed Gini coefficient for 16,134 genes using 8,556 GTEx libraries [55]. Gini coefficient measures statistical dispersion which can be used to measure gene expression ubiquity: value 0 represents genes expressed in all samples, while value 1 represents genes expressed in only one sample. We observed that the correlations obtained between Gini coefficient and model errors in each TCGA sample ranged from 0.22 to 0.36. We also compared model errors associated to first and last quartiles of the Gini coefficient distribution using a Wilcoxon test for each of the 241 samples. The test was invariably significant with maximum p-value = $2.881e^{-7}$. Likewise analyses were performed with 1,897 FANTOM CAGE libraries [56] considering 15,904 genes. In that case, correlation between models errors and Gini coefficients ranged from 0.25 to 0.4. Overall these analyses suggested that our model better predicts expression of highly and ubiquitously expressed genes. We do not exclude that, when predicting tissue-specific genes, ChIP-seq data collected from the same tissue may add explanatory power to the sequence model. Note, however, that the model performances vary between cancer and cell types implying that part of cell-specific genes are also well predicted by the model (S9 Fig).

Relationships between selected nucleotide composition and genome architecture

We probed the regulatory activities of the selected regions. We first determined whether introns contained specific regulatory sequence code by assessing the presence of *cis* expression

quantitative trait loci (*cis*-eQTLs). Zhou *et al.* indeed showed that the effect of eQTL SNPs can be predicted from a regulatory sequence code learned from genomic sequences [25]. These findings also implied that *cis*-eQTLs preferentially affect DNA sequences at precise locations (e.g. TF binding sites) rather than global nucleotide composition (i.e. nucleotide/dinucleotide percentages used as variables in our model). We used the v6p GTEx release to compute the average frequencies of *cis*-eQTLs present in the considered genomic regions and directly linked to their host genes (S6 Table). We noticed that introns contained the smallest density of *cis*-eQTLs (10 times less than any other regions), while containing comparable amount of SNPs (S7 Table). This result argued against the presence of a regulatory sequence code similar to that observed in promoters for instance [25], despite the presence of enhancers (S8 Table. These results rather unveiled the existence of another layer of intron-mediated regulation, which involves global nucleotide compositions of larger DNA regions. We then asked whether the groups of genes identified by the regression trees (Fig 6) correspond to specific TADs. Genes within the same TAD tend to be coordinately expressed [57, 58]. TADs with similar chromatin states tend to associate to form two genomic compartments called A and B: A contains transcriptionally active regions while B corresponds to transcriptionally inactive regions [59]. The driving forces behind this compartmentalization and the transitions between compartments observed in different cell types are not fully understood, but chromatin composition and transcription are supposed to play key roles [5]. Jabbari and Bernardi showed that nucleotide composition along the genome (notably isochores) can help define TADs [60]. As intronic sequences represent $\sim$ 50% of the human genome (1,512,685,844 bp out of 3,137,161,264 according to ENSEMBL merged intron coordinates), the nucleotide composition of introns likely resemble that of neighbor genes and more globally that of the corresponding TAD. We used the 373 TADs containing more than 10 genes mapped in IMR90 cells [6]. For each TAD and each (di)nucleotide, we used a Kolmogorov-Smirnov test to compare the (di)nucleotide distribution of the embedded genes with that of all other genes. We used a Benjamini-Hochberg multiple testing correction to control the False Discovery Rate (FDR), which was fixed at 0.05 (see Materials and methods section). We found that 324 TADs out of 373 ($\sim$87%) are characterized by at least one specific nucleotide signature (Fig 7A). In addition, our results clearly showed the existence of distinct classes of TADs related to GC content (GC-rich, GC-poor and intermediate GC content) (Fig 7A), in agreement with [60]. We next considered the 967 groups of genes defined in Fig 6 whose expression is accurately predicted by our model (i.e. groups with mean error $<$ mean error of the 1st quartile). We thus focused our analyses on genes for which we did learn some regulatory features. We evaluated the enrichment for specific TADs in each group (considering only TADs containing more than 10 genes) using an hypergeometric test (Fig 7B). We found that 60% of these groups were enriched for at least one TAD (p-value $<$ 0.05). Hence, several groups of genes identified by the regression trees (Fig 6) do correspond to specific TADs (Fig 7B). We concluded that our model, primarily based on intronic sequences, select gene nucleotide compositions that better distinguish active TADs.

Discussion

In this study, we corroborate the hypothesis that DNA sequence contains information able to explain gene expression [20–25]. We built a global regression model to predict, in any given sample, the expression of the different genes using only nucleotide compositions as predictive variables. Overall our model provided a framework to study gene regulation, in particular the influence of regulatory regions and their associated nucleotide composition.

A surprising result of our study is that sequence-level information is highly predictive of gene expression and in some occasions comparable to reference ChIP-seq data alone [17, 19].

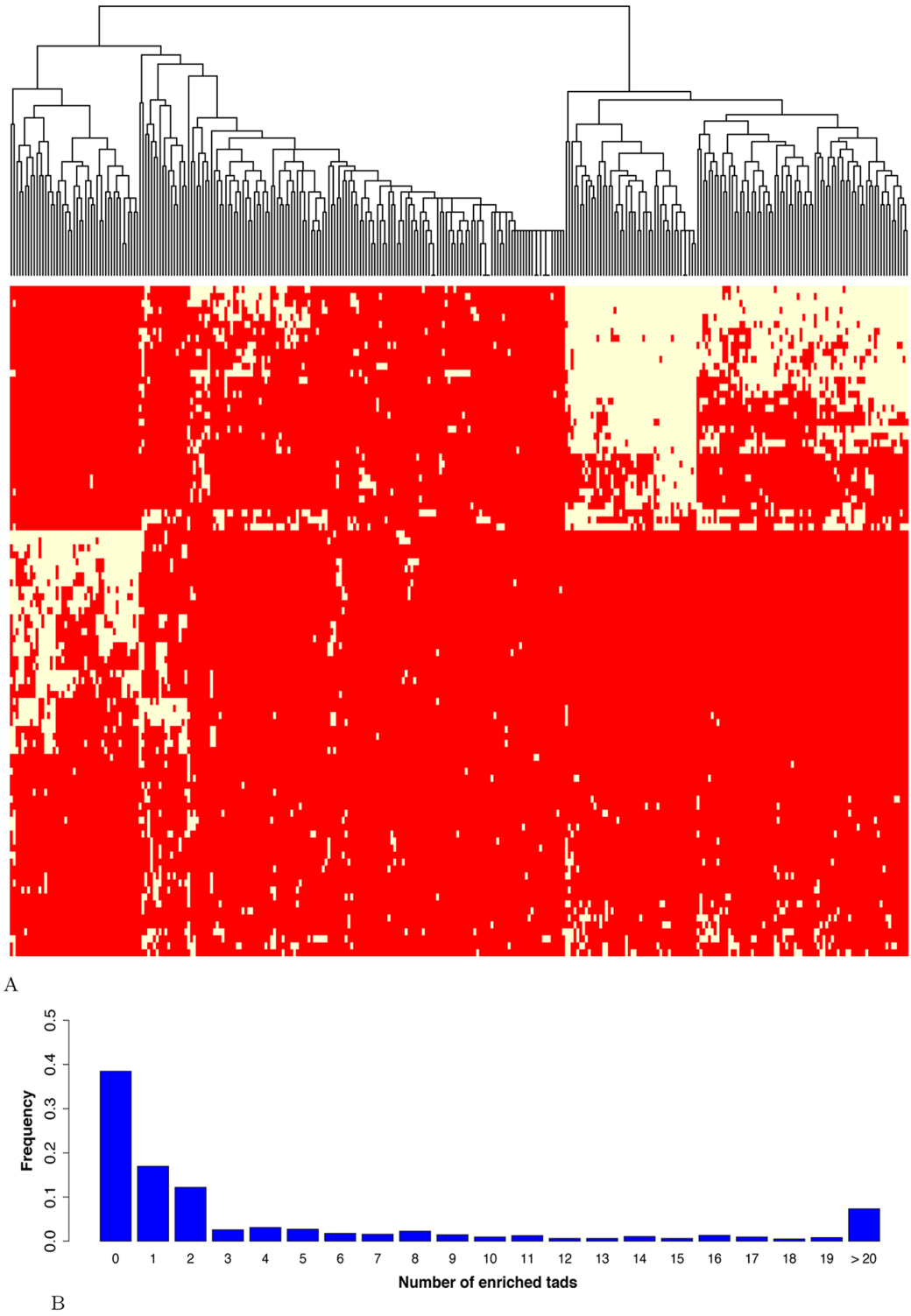


Fig 7. A: Nucleotide compositions of resident genes distinguish TADs. For each TAD and for each region considered, the percentage of each nucleotide and dinucleotide associated to the embedded genes were compared to that of all other genes using a Kolmogorov-Smirnov test. Red indicates FDR-corrected p-value ≥ 0.05 and yellow FDR-corrected p-value < 0.05. TAD clustering was made using this binary information. Only TADs with at least one p-value < 0.05 are shown (i.e. 87% of the TADs containing at least 10 genes). y-axis from top to bottom: G_INTR, GpC_INTR, CpC_INTR, CpC_3UTR,

GpC_3UTR, G_3UTR, GpC_CDS, CpC_CDS, G_CDS, G_DFR, CpC_DFR, GpC_DFR, CpG_INTR, CpG_3UTR, CpG_CDS, CpG_DFR, G_DU, GpC_DD, CpG, DU, CpG_DD, GpC_DU, CpC_DU, CpC_DD, G_DD, GpC_5UTR, CpG_5UTR, G_5UTR, GpC_CORE, CpG_CORE, CpC_CORE, G_CORE, CpC_5UTR, CpT_3UTR, CpT_CDS, CpT_INTR, ApT_INTR, TpA_INTR, A_INTR, ApA_INTR, TpA_3UTR, ApT_3UTR, A_3UTR, ApA_3UTR, ApA_CDS, A_CDS, ApT_CDS, TpA_CDS, A_DD, ApA_DD, ApT_DD, TpA_DD, TpA_DU, ApT_DU, ApA_DU, A_DU, TpA_DFR, ApT_DFR, A_DFR, ApA_DFR, ApA_CORE, A_CORE, ApT_CORE, TpA_CORE, ApA_5UTR, ApT_5UTR, A_5UTR, TpA_5UTR, ApC_DFR, ApC_DD, ApC_DU, TpC_DU, TpC_DFR, ApC_CORE, CpA_DU, CpA_DFR, CpA_CDS, ApC_CDS, ApC_3UTR, TpC_CDS, TpC_CORE, CpT_5UTR, TpC_5UTR, CpT_CORE, TpC_DD, CpA_CORE, ApC_5UTR, CpA_5UTR, ApC_INTR, CpA_DD, CpT_DFR, CpT_DD, CpT_DU, TpC_3UTR, TpC_INTR, CpA_INTR, CpA_3UTR. **B: TAD enrichment within groups of genes whose expression is accurately predicted by our model.** The enrichment for each TAD (containing more than 10 genes) in each gene group accurately predicted by our model (i.e. groups with mean error < mean errors of the 1st quartile) was evaluated using an hypergeometric test. The fraction of groups with enriched TADs (p-value < 0.05) is represented.

https://doi.org/10.1371/journal.pcbi.1005921.g007

The similar accuracy of models built on real and randomly permuted experimental data indicated that, though the experimental data are biologically relevant, their interpretation through a linear model, in particular inference of TF combinations, is not straightforward as randomization of experimental data did not show the expected loss of accuracy (Fig 3). An interesting perspective would be to devise a strategy to infer TF combinations from experimental data without being influenced by the opening of the chromatin.

The accuracy of our model confirmed that DNA sequence *per se* and basic information like dinucleotide frequencies have very high predictive power. It remains to determine the exact nature of these sequence-level instructions. Interestingly, nucleotide environment contributes to prediction of TF binding sites and motifs bound by a TF have a unique sequence environment that resembles the motif itself [40]. Hence, the potential of the nucleotide content to predict gene expression may be related to the presence of regulatory motifs and TFBSs. However, we showed that the gene body (introns, CDS and UTRs), as opposed to sequences located upstream (promoter) or downstream (DFR), had the most significant contribution in our model. Moreover, *cis*-eQTL frequencies argue against the presence of a regulatory sequence code in introns similar to that observed in promoters, suggesting the existence of another layer of regulation implicating the nucleotide composition of large DNA regions.

Gene nucleotide compositions vary across the genome and can even help define TAD boundaries [60]. In line with [60], we showed that genes located within the same TAD share similar nucleotide compositions, which provides a nucleotide signature for their TADs (Fig 7A). Our model aimed at predicting gene expression, and therefore intimately linked to TAD compartmentalization, appeared to capture these signatures. Several studies have already demonstrated the existence of sequence-level instructions able to determine genomic interactions. Using an SVM-based approach, Nikumbh *et al* demonstrated that sequence features can determine long-range chromosomal interactions [61]. Similar results were obtained by Singh *et al.* using deep learning-based models [62]. Using biophysical approaches, Kornyshev *et al.* showed that sequence homology influences physical attractive forces between DNA fragments [63]. It would be interesting to determine whether the nucleotide signatures identified by our model are directly implicated in DNA folding and 3D genome architecture.

Finally, although sequence-level instructions are—almost—identical in all cells of an individual, their usage must be cell-type specific to allow proper A/B compartimentalization of TADs, gene expression and ultimately diversity of cell functions. At this stage, the mechanisms driving this cell-type specific selection of nucleotide compositions remain to be characterized.

Supporting information

S1 Fig. Comparison of models built on maximum or sum PWM motif scores. The model was built (i) using 60 nucleotide/dinucleotide percentages computed in the 3 promoter

segments (CORE+DU+DD) and 471 JASPAR2016 PWM maximum scores computed in the CORE segment (pink) or (ii) using 60 nucleotide/dinucleotide percentages computed in the 3 promoter segments (CORE+DU+DD) and 471 JASPAR2016 PWM sum scores computed in the CORE segment (green). All sequences were centered around the 2nd TSS and the 2 models were fitted on 16,294 genes for each of the 241 samples.
(PDF)

S2 Fig. Dinucleotide local distribution around GENCODEv24 TSSs. Dinucleotide percentages (y-axis) along 140,604 DNA regions centered around GENCODE v24 TSSs ±2000 bp (the distance to TSS is shown in the x-axis). Dinucleotide combinations are represented as first nucleotide on left and second nucleotide on top. The promoter segmentation used in this study (Fig 1) is indicated with vertical dashed lines at -500 bp and 500 bp from the TSS.
(PDF)

S3 Fig. Number of TSSs by gene. We considered 19,393 TCGA genes listed in TCGA and the TSSs annotated by GENCODE v24.
(PDF)

S4 Fig. Contribution in the model of the TSS number. The model is built using 20 variables corresponding to the nucleotide (4) and dinucleotide (16) percentages computed in the CORE promoter (red), DU (green) or DD (yellow) centered around the second TSS as predictive variables (green). Linear models are also built on the number of isoforms (dark pink) and the number of TSSs (dark blue). Finally models are built using the combinations of variables indicated. All different models were fitted on 19,393 genes for each of the 241 samples considered. The prediction accuracy was evaluated in each sample by evaluating the Spearman correlation coefficients between observed and predicted gene expressions. The correlations obtained in all samples are shown as violin plots. These two last plots underscored the importance of these two variables in predicting gene expression.
(PDF)

S5 Fig. Gene expression distribution and FANTOM5 enhancer association. The 19,393 genes listed in one LAML sample (TCGA.AB.2939.03A.01T.0740.13_LAML) (pink) and a subset of 11,359 genes with assigned FANTOM enhancers (green) were considered. The median expression of genes with assigned enhancers is greater than that of all genes (wilcoxon test p-value $<$ 2.2e-16)
(PDF)

S6 Fig. Accuracies of models built on dsDNA or ssDNA. A: Models were built using nucleotide and dinucleotide percentages computed on dsDNA (2 nucleotides + 8 dinucleotides; green violin) or on ssDNA (4 nucleotides + 16 dinucleotides; purple violin) in all the regulatory regions (CORE, DU, DD, 5UTR, CDS, 3UTR, INTR, DFR). The 2 models were fitted on 16,294 genes for each of the 241 samples. The prediction accuracy was evaluated in each sample by evaluating the Spearman correlation coefficients. **B:** Same analyses focusing on each of the indicated regions.
(PDF)

S7 Fig. Model accuracy with different set of nucleotide predictive variables. A: Models were built using different set of variables including nucleotide (4 x 8 regions), dinucleotide (16 x 8 regions) and/or trinucleotide (64 x 8 regions) percentages computed in all the regulatory regions (CORE, DU, DD, 5UTR, CDS, 3UTR, INTR, DFR). All different models were fitted on 16,280 genes for each of the 241 samples considered. The prediction accuracy was evaluated in each sample by evaluating the Spearman correlation coefficients. **B:** Models were built using

nucleotide (4 x 8 regions) and dinucleotide (16 x 8 regions) percentages computed in all the regulatory regions and trinucleotide (64) percentages computed in each of the indicated region separately.
(PDF)

S8 Fig. Forward selection procedure with models built on isoform expressions. The procedure is identical to that described in Fig 4 but models were built on isoform-specific variables and correlations were computed between observed and predicted isoform expression, not gene expression.
(PDF)

S9 Fig. Model accuracy in different cancer types. The model with 160 variables (20 (di)nucleotide rates in 8 regions) was built on 16,294 genes in 241 samples corresponding to the initial training set corresponding to 12 cancer types (**A**) and in an additional set of 1,270 samples corresponding to 14 different cancer types (**B**). The prediction accuracy was evaluated in each sample by evaluating the Spearman correlation coefficients between observed and predicted gene expressions. The correlations obtained in all samples of each data sets are shown as violin plots in **A** (training set) and **B** (additional set). The color code indicates the cancer types. The horizontal dashed lines indicates the median correlation (**A**, 0.582; **B**, 0.577).
(PDF)

S10 Fig. Comparison on models built on RNA-seq or microarray data. The model with 160 variables (20 (di)nucleotide rates in 8 regions) was built on 9,791 genes in 582 samples with matched RNA-seq and microarray data. The prediction accuracy was evaluated in each sample by evaluating the Spearman correlation coefficients between observed and predicted gene expressions. The correlations obtained in all samples with RNA-seq- or microarray-built models are shown as violin plots.
(PDF)

S11 Fig. Spearman correlations between CNV segment mean score and model prediction error. CNV absolute segment mean scores were computed for each as explained in Materials and Methods section. Model prediction absolute error for each gene are given by our predictive model using nucleotide and dinuclotide percentages computed in all the regulatory regions. Models were fitted on 16,294 genes for each of the 234 on 241 samples having CNV TCGA data available. The median correlation for the 234 samples is -0.014.
(PDF)

S12 Fig. Absolute values of the regression coefficients. A linear regression model was built, for each sample, on standardized stable variables only. The boxplots show absolute values of the corresponding coefficients in all samples for each variable considered. Color code as in Fig 5. CpG in the core promoter is highlighted in white. Purple line represents the median of CpG_CORE coefficients.
(PDF)

S13 Fig. Example of regression trees learned on two linear models. A: Regression tree leading to a group of genes well predicted in all samples. This tree has been learned on the sample TCGA.FC.A5OB.01A.11R.A29R.07_PRAD using all nucleotide composition in all regions. The red path defines a group of 996 genes which has low Lasso error in all samples and cancer types. This group was used for functional annotation (S4 Table). **B: Regression tree leading to a group of genes well predicted in LGG and PPAD samples.** This tree has been learned on the sample TCGA.IB.7646.01A.11R.2156.07_PAAD using all nucleotide composition in all

regions. The red path defines a group of 1,531 genes which has low Lasso error in LGG and PAAD samples but high error in LAML, LIHC and DLBC samples. This group was used for functional annotation (S5 Table).
(PDF)

S1 Table. Model comparison. Each model is fitted for each tumor, using all the variables over all regions (160 variables among 8 regulatory regions). First and second columns are median correlation and mean square error over all the tumors. The third column represents mean computing time per tumor (in minutes) on a standard laptop.
(PDF)

S2 Table. Contributions of additional genomic regions. Genomic regions were ranked according to their contribution in predicting gene expression. First, all regions were tested separately. Introns yielded the highest Spearman correlation between observed and predicted expressions and was selected as the 'first' seed region. Second, each region not already in the model was added separately. 5UTR in association with introns yielded the best correlation and was therefore selected as the 'second' region. Third, the procedure was repeated till all regions were included in the model. The contribution of each region is then visualized starting from the most important (left) to the less important (right). The correlations computed at each steps are indicated.
(PDF)

S3 Table. Correlations between observed and predicted isoform expression. The procedure is identical to that described in S2 Table but models were built on isoform-specific variables and correlations were computed between observed and predicted isoform expression, not gene expression.
(PDF)

S4 Table. Functional enrichment of a group of genes well predicted in all samples. The group of 996 genes is obtained by fitting a regression tree on the sample TCGA.FC.A5OB.01A.11R.A29R.07_PRAD using all the nucleotide composition in all regions. These genes are well predicted (mean error $<$ 1st quartile) for all samples of different type cancers. This group of genes was further annotated using the DAVID functional annotation tool. Only the top 5 biological processes indicated by DAVID is shown. The GO term yielded by this analysis corresponded to general and widespread biological processes indicating that these genes likely corresponded to housekeeping genes.
(PDF)

S5 Table. Functional enrichment of a group of genes well predicted in LGG and PAAD. The group of 1,531 genes is obtained by fitting a regression tree on the sample TCGA.IB.7646.01A.11R.2156.07_PAAD using all the nucleotide composition in all regions. These genes are well predicted (mean error $<$ 1st quartile) for all LGG and PAAD samples but not that of LAML, DBLC and LIHC. This group of genes was further annotated using the DAVID functional annotation tool. Only the top 5 biological processes indicated by DAVID is shown. The GO term "Nervous system development" indicates that these genes can be involved in specific biological processes.
(PDF)

S6 Table. Frequencies of *cis*-eQTLs in the genomic regions considered. We computed the density of *cis*-eQTL per regulatory region by dividing the sum of *cis*-eQTLs intersecting with the region considered for all genes by the sum of the lengths of the same regulatory region of

all genes. see Material and methods for details.
(PDF)

S7 Table. Frequencies of SNPs in CORE and INTRON regions. We computed the density of SNPs per regulatory region by dividing the sum of SNPs intersecting with the region considered for all genes by the sum of the lengths of the same regulatory region of all genes. We only considered SNPs detected on chromosomes 1, 2 and 19. see Material and methods for details.
(PDF)

S8 Table. Intersection between enhancers and the genomic regions considered. We computed the density of enhancers per regulatory region by dividing the total length of the intersection between the enhancers and the region considered for all genes by the sum of the lengths of the same regulatory region of all genes. see Material and methods for details.
(PDF)

Acknowledgments

We thank Mohamed Elati, Mathieu Lajoie, Anthony Mathelier and Cédric Notredame for insightful discussions and suggestions. We also thank Yue Li, Zhaolei Zhang, Florian Schmidt and Marcel H. Schulz for sharing data. We are indebted to the researchers around the globe who generated experimental data and made them freely available. C-H.L. is grateful to Marc Piechaczyk, Edouard Bertrand, Anthony Mathelier and Wyeth W. Wasserman for continued support.

Author Contributions

Conceptualization: Laurent Bréhélin, Sophie Lèbre, Charles-Henri Lecellier.

Formal analysis: Chloé Bessière, May Taha, Florent Petitprez, Jimmy Vandel.

Funding acquisition: Jean-Michel Marin, Laurent Bréhélin, Sophie Lèbre, Charles-Henri Lecellier.

Investigation: Chloé Bessière, May Taha, Florent Petitprez, Jimmy Vandel, Laurent Bréhélin, Sophie Lèbre, Charles-Henri Lecellier.

Methodology: Laurent Bréhélin, Sophie Lèbre, Charles-Henri Lecellier.

Project administration: Laurent Bréhélin, Sophie Lèbre, Charles-Henri Lecellier.

Supervision: Laurent Bréhélin, Sophie Lèbre, Charles-Henri Lecellier.

Validation: Chloé Bessière, May Taha.

Writing – original draft: Laurent Bréhélin, Sophie Lèbre, Charles-Henri Lecellier.

Writing – review & editing: Chloé Bessière, May Taha, Florent Petitprez, Jimmy Vandel, Jean-Michel Marin, Laurent Bréhélin, Sophie Lèbre, Charles-Henri Lecellier.

References

1. Andersson R, Sandelin A, Danko CG. A unified architecture of transcriptional regulatory elements. Trends in genetics: TIG. 2015; 31(8):426–433. https://doi.org/10.1016/j.tig.2015.05.007 PMID: 26073855
2. Babu D, Fullwood MJ. 3D genome organization in health and disease: emerging opportunities in cancer translational medicine. Nucleus (Austin, Tex). 2015; 6(5):382–393.

3. Ea V, Baudement MO, Lesne A, Forné T. Contribution of Topological Domains and Loop Formation to 3D Chromatin Organization. Genes. 2015; 6(3):734–750. https://doi.org/10.3390/genes6030734 PMID: 26226004
4. Gonzalez-Sandoval A, Gasser SM. On TADs and LADs: Spatial Control Over Gene Expression. Trends Genet. 2016;. https://doi.org/10.1016/j.tig.2016.05.004 PMID: 27312344
5. Merkenschlager M, Nora EP. CTCF and Cohesin in Genome Folding and Transcriptional Gene Regulation. Annu Rev Genomics Hum Genet. 2016; 17:17–43. https://doi.org/10.1146/annurev-genom-083115-022339 PMID: 27089971
6. Dixon JR, Selvaraj S, Yue F, Kim A, Li Y, Shen Y, et al. Topological domains in mammalian genomes identified by analysis of chromatin interactions. Nature. 2012; 485(7398):376–380. https://doi.org/10.1038/nature11082 PMID: 22495300
7. Andersson R, Gebhard C, Miguel-Escalada I, Hoof I, Bornholdt J, Boyd M, et al. An atlas of active enhancers across human cell types and tissues. Nature. 2014; 507(7493):455–461. https://doi.org/10.1038/nature12787 PMID: 24670763
8. Johnson DS, Mortazavi A, Myers RM, Wold B. Genome-wide mapping of in vivo protein-DNA interactions. Science. 2007; 316(5830):1497–1502. https://doi.org/10.1126/science.1141319 PMID: 17540862
9. Slattery M, Riley T, Liu P, Abe N, Gomez-Alcala P, Dror I, et al. Cofactor binding evokes latent differences in DNA binding specificity between Hox proteins. Cell. 2011; 147(6):1270–1282. https://doi.org/10.1016/j.cell.2011.10.053 PMID: 22153072
10. Ray D, Kazan H, Chan ET, Pena Castillo L, Chaudhry S, Talukder S, et al. Rapid and systematic analysis of the RNA recognition specificities of RNA-binding proteins. Nat Biotechnol. 2009; 27(7):667–670. https://doi.org/10.1038/nbt.1550 PMID: 19561594
11. Wasserman WW, Sandelin A. Applied bioinformatics for the identification of regulatory elements. Nat Rev Genet. 2004; 5(4):276–287. https://doi.org/10.1038/nrg1315 PMID: 15131651
12. Ravasi T, Suzuki H, Cannistraci CV, Katayama S, Bajic VB, Tan K, et al. An atlas of combinatorial transcriptional regulation in mouse and man. Cell. 2010; 140(5):744–752. https://doi.org/10.1016/j.cell.2010.01.044 PMID: 20211142
13. Gerstberger S, Hafner M, Tuschl T. A census of human RNA-binding proteins. Nat Rev Genet. 2014; 15(12):829–845. https://doi.org/10.1038/nrg3813 PMID: 25365966
14. Dunham I, Kundaje A, Aldred SF, Collins PJ, Davis CA, Doyle F, et al. An integrated encyclopedia of DNA elements in the human genome. Nature. 2012; 489(7414):57–74. https://doi.org/10.1038/nature11247
15. Lundberg SM, Tu WB, Raught B, Penn LZ, Hoffman MM, Lee SI. ChromNet: Learning the human chromatin network from all ENCODE ChIP-seq data. Genome Biol. 2016; 17:82. https://doi.org/10.1186/s13059-016-0925-0 PMID: 27139377
16. Cheng C, Alexander R, Min R, Leng J, Yip KY, Rozowsky J, et al. Understanding transcriptional regulation by integrative analysis of transcription factor binding data. Genome Res. 2012; 22(9):1658–1667. https://doi.org/10.1101/gr.136838.111 PMID: 22955978
17. Li Y, Liang M, Zhang Z. Regression analysis of combined gene expression regulation in acute myeloid leukemia. PLoS Comput Biol. 2014; 10(10):e1003908. https://doi.org/10.1371/journal.pcbi.1003908 PMID: 25340776
18. Jiang P, Freedman ML, Liu JS, Liu XS. Inference of transcriptional regulation in cancers. Proc Natl Acad Sci USA. 2015; 112(25):7731–7736. https://doi.org/10.1073/pnas.1424272112 PMID: 26056275
19. Schmidt F, Gasparoni N, Gasparoni G, Gianmoena K, Cadenas C, Polansky JK, et al. Combining transcription factor binding affinities with open-chromatin data for accurate gene expression prediction. Nucleic Acids Res. 2017; 45(1):54–66. https://doi.org/10.1093/nar/gkw1061 PMID: 27899623
20. Quante T, Bird A. Do short, frequent DNA sequence motifs mould the epigenome? Nat Rev Mol Cell Biol. 2016; 17(4):257–262. PMID: 26837845
21. McVicker G, van de Geijn B, Degner JF, Cain CE, Banovich NE, Raj A, et al. Identification of genetic variants that affect histone modifications in human cells. Science. 2013; 342(6159):747–749. https://doi.org/10.1126/science.1242429 PMID: 24136359
22. Kilpinen H, Waszak SM, Gschwind AR, Raghav SK, Witwicki RM, Orioli A, et al. Coordinated effects of sequence variation on DNA binding, chromatin structure, and transcription. Science. 2013; 342(6159):744–747. https://doi.org/10.1126/science.1242463 PMID: 24136355
23. Kasowski M, Kyriazopoulou-Panagiotopoulou S, Grubert F, Zaugg JB, Kundaje A, Liu Y, et al. Extensive variation in chromatin states across humans. Science. 2013; 342(6159):750–752. https://doi.org/10.1126/science.1242510 PMID: 24136358

24. Whitaker JW, Chen Z, Wang W. Predicting the human epigenome from DNA motifs. Nat Methods. 2015; 12(3):265–272. https://doi.org/10.1038/nmeth.3065 PMID: 25240437

25. Zhou J, Troyanskaya OG. Predicting effects of noncoding variants with deep learning-based sequence model. Nat Methods. 2015; 12(10):931–934. https://doi.org/10.1038/nmeth.3547 PMID: 26301843

26. Raghava GP, Han JH. Correlation and prediction of gene expression level from amino acid and dipeptide composition of its protein. BMC Bioinformatics. 2005; 6:59. https://doi.org/10.1186/1471-2105-6-59 PMID: 15773999

27. Quinlan AR. BEDTools: The Swiss-Army Tool for Genome Feature Analysis. Curr Protoc Bioinformatics. 2014; 47:1–34.

28. Mathelier A, Fornes O, Arenillas DJ, Chen CY, Denay G, Lee J, et al. JASPAR 2016: a major expansion and update of the open-access database of transcription factor binding profiles. Nucleic Acids Res. 2016; 44(D1):D110–115. https://doi.org/10.1093/nar/gkv1176 PMID: 26531826

29. Chiu TP, Comoglio F, Zhou T, Yang L, Paro R, Rohs R. DNAshapeR: an R/Bioconductor package for DNA shape prediction and feature encoding. Bioinformatics. 2016; 32(8):1211–1213. https://doi.org/10.1093/bioinformatics/btv735 PMID: 26668005

30. Jiao X, Sherman BT, Huang daW, Stephens R, Baseler MW, Lane HC, et al. DAVID-WS: a stateful web service to facilitate gene/protein list analysis. Bioinformatics. 2012; 28(13):1805–1806. https://doi.org/10.1093/bioinformatics/bts251 PMID: 22543366

31. Tibshirani R. Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society Series B (Methodological). 1996; p. 267–288.

32. R Core Team. R: A Language and Environment for Statistical Computing; 2013. Available from: http://www.R-project.org/.

33. Breiman L, Friedman J, Olshen R, Stone C. Classification and Regression Trees. Monterey, CA: Wadsworth and Brooks; 1984.

34. Breiman L. Random Forests. Machine Learning. 2001; 45(1):5–32. https://doi.org/10.1023/A:1010933404324

35. Meinshausen N, Bühlmann P. Stability selection. Journal of the Royal Statistical Society: Series B (Statistical Methodology). 2010; 72(4):417–473. https://doi.org/10.1111/j.1467-9868.2010.00740.x

36. Sill M, Hielscher T, Becker N, Zucknick M, et al. c060: Extended inference with lasso and elastic-net regularized Cox and generalized linear models. Journal of Statistical Software. 2015; 62(5).

37. Benjamini Y, Hochberg Y. Controlling the false discovery rate: a practical and powerful approach to multiple testing. Journal of the royal statistical society Series B (Methodological). 1995; p. 289–300.

38. Lenhard B, Sandelin A, Carninci P. Metazoan promoters: emerging characteristics and insights into transcriptional regulation. Nat Rev Genet. 2012; 13(4):233–245. PMID: 22392219

39. Nguyen TA, Jones RD, Snavely A, Pfenning A, Kirchner R, Hemberg M, et al. High-throughput functional comparison of promoter and enhancer activities. Genome Res. 2016;. https://doi.org/10.1101/gr.204834.116

40. Dror I, Golan T, Levy C, Rohs R, Mandel-Gutfreund Y. A widespread role of the motif environment in transcription factor binding across diverse protein families. Genome Res. 2015; 25(9):1268–1280. https://doi.org/10.1101/gr.184671.114 PMID: 26160164

41. Diamanti K, Umer HM, Kruczyk M, Dąbrowski MJ, Cavalli M, Wadelius C, et al. Maps of context-dependent putative regulatory regions and genomic signal interactions. Nucleic Acids Res. 2016;. https://doi.org/10.1093/nar/gkw800 PMID: 27625394

42. Ray D, Kazan H, Cook KB, Weirauch MT, Najafabadi HS, Li X, et al. A compendium of RNA-binding motifs for decoding gene regulation. Nature. 2013; 499(7457):172–177. https://doi.org/10.1038/nature12311 PMID: 23846655

43. Li X, Quon G, Lipshitz HD, Morris Q. Predicting in vivo binding sites of RNA-binding proteins using mRNA secondary structure. RNA. 2010; 16(6):1096–1107. https://doi.org/10.1261/rna.2017210 PMID: 20418358

44. Auweter SD, Oberstrass FC, Allain FH. Sequence-specific binding of single-stranded RNA: is there a code for recognition? Nucleic Acids Res. 2006; 34(17):4943–4959.

45. Liu C, Mallick B, Long D, Rennie WA, Wolenc A, Carmack CS, et al. CLIP-based prediction of mammalian microRNA binding sites. Nucleic Acids Res. 2013; 41(14):e138. https://doi.org/10.1093/nar/gkt435 PMID: 23703212

46. Boel G, Letso R, Neely H, Price WN, Wong KH, Su M, et al. Codon influence on protein expression in E. coli correlates with mRNA levels. Nature. 2016; 529(7586):358–363. https://doi.org/10.1038/nature16509 PMID: 26760206

47. Bazzini AA, Del Viso F, Moreno-Mateos MA, Johnstone TG, Vejnar CE, Qin Y, et al. Codon identity regulates mRNA stability and translation efficiency during the maternal-to-zygotic transition. EMBO J. 2016;. https://doi.org/10.15252/embj.201694699

48. Presnyak V, Alhusaini N, Chen YH, Martin S, Morris N, Kline N, et al. Codon optimality is a major determinant of mRNA stability. Cell. 2015; 160(6):1111–1124. https://doi.org/10.1016/j.cell.2015.02.029 PMID: 25768907

49. Chorev M, Carmel L. The function of introns. Front Genet. 2012; 3:55. https://doi.org/10.3389/fgene.2012.00055 PMID: 22518112

50. Rose AB. Intron-mediated regulation of gene expression. Curr Top Microbiol Immunol. 2008; 326:277–290. PMID: 18630758

51. Schwalb B, Michel M, Zacher B, Fruhauf K, Demel C, Tresch A, et al. TT-seq maps the human transient transcriptome. Science. 2016; 352(6290):1225–1228. https://doi.org/10.1126/science.aad9841 PMID: 27257258

52. Bunting KL, Soong TD, Singh R, Jiang Y, Beguelin W, Poloway DW, et al. Multi-tiered Reorganization of the Genome during B Cell Affinity Maturation Anchored by a Germinal Center-Specific Locus Control Region. Immunity. 2016; 45(3):497–512. https://doi.org/10.1016/j.immuni.2016.08.012 PMID: 27637145

53. Hayer KE, Pizarro A, Lahens NF, Hogenesch JB, Grant GR. Benchmark analysis of algorithms for determining and quantifying full-length mRNA splice forms from RNA-seq data. Bioinformatics. 2015; 31(24):3938–3945. https://doi.org/10.1093/bioinformatics/btv488 PMID: 26338770

54. Breiman L, et al. Classification and Regression Trees. New York: Chapman & Hall; 1984.

55. Mele M, Ferreira PG, Reverter F, DeLuca DS, Monlong J, Sammeth M, et al. Human genomics. The human transcriptome across tissues and individuals. Science. 2015; 348(6235):660–665. https://doi.org/10.1126/science.aaa0355 PMID: 25954002

56. Forrest AR, Kawaji H, Rehli M, Baillie JK, de Hoon MJ, Haberle V, et al. A promoter-level mammalian expression atlas. Nature. 2014; 507(7493):462–470. https://doi.org/10.1038/nature13182 PMID: 24670764

57. Nora EP, Lajoie BR, Schulz EG, Giorgetti L, Okamoto I, Servant N, et al. Spatial partitioning of the regulatory landscape of the X-inactivation centre. Nature. 2012; 485(7398):381–385. https://doi.org/10.1038/nature11049 PMID: 22495304

58. Fanucchi S, Shibayama Y, Burd S, Weinberg MS, Mhlanga MM. Chromosomal contact permits transcription between coregulated genes. Cell. 2013; 155(3):606–620. https://doi.org/10.1016/j.cell.2013.09.051 PMID: 24243018

59. Lieberman-Aiden E, Berkum NLv, Williams L, Imakaev M, Ragoczy T, Telling A, et al. Comprehensive Mapping of Long-Range Interactions Reveals Folding Principles of the Human Genome. Science. 2009; 326(5950):289–293. https://doi.org/10.1126/science.1181369 PMID: 19815776

60. Jabbari K, Bernardi G. An Isochore Framework Underlies Chromatin Architecture. PLoS ONE. 2017; 12(1):e0168023. https://doi.org/10.1371/journal.pone.0168023 PMID: 28060840

61. Nikumbh S, Pfeifer N. Genetic sequence-based prediction of long-range chromatin interactions suggests a potential role of short tandem repeat sequences in genome organization. BMC Bioinformatics. 2017; 18(1):218. https://doi.org/10.1186/s12859-017-1624-x PMID: 28420341

62. Singh S, Yang Y, Poczos B, Ma J. Predicting Enhancer-Promoter Interaction from Genomic Sequence with Deep Neural Networks. BioRxiv. 2016;

63. Kornyshev AA, Leikin S. Sequence recognition in the pairing of DNA duplexes. Phys Rev Lett. 2001; 86(16):3666–3669. https://doi.org/10.1103/PhysRevLett.86.3666 PMID: 11328049

SUPPORTING INFORMATION

Probing instructions for expression regulation in gene nucleotide compositions

Chloé Bessière [1,2☯], May Taha [1,2,3☯], Florent Petitprez [1,2], Jimmy Vandel [1,4], Jean-Michel Marin [1,3], Sophie Lèbre [1,3*], Laurent Bréhélin [1,4*], Charles-Henri Lecellier [1,2*]

1 IBC, CNRS, Univ. Montpellier, France
2 Institut de Génétique Moléculaire de Montpellier, University of Montpellier, CNRS, Montpellier, France
3 IMAG, Univ. Montpellier, France
4 LIRMM, Univ. Montpellier, France

☯These authors contributed equally to this work.
* sophie.lebre@umontpellier.fr, brehelin@lirmm.fr, charles.lecellier@igmm.cnrs.fr

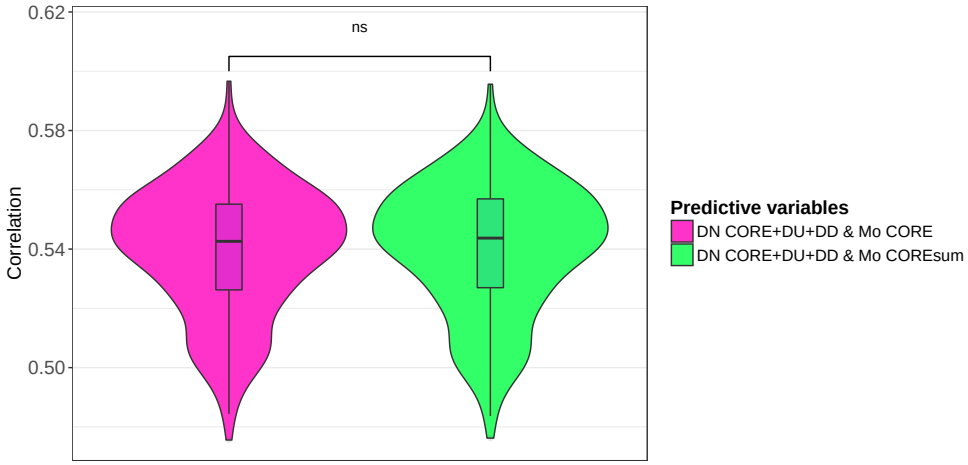


Fig S1

Method	Median correlation	Median error	Mean time
Linear model (with Lasso)	0.578	6.50	0.13
Random Forest	0.592	6.293	5.033
Regression trees	0.45	7.456	0.172

Table S1

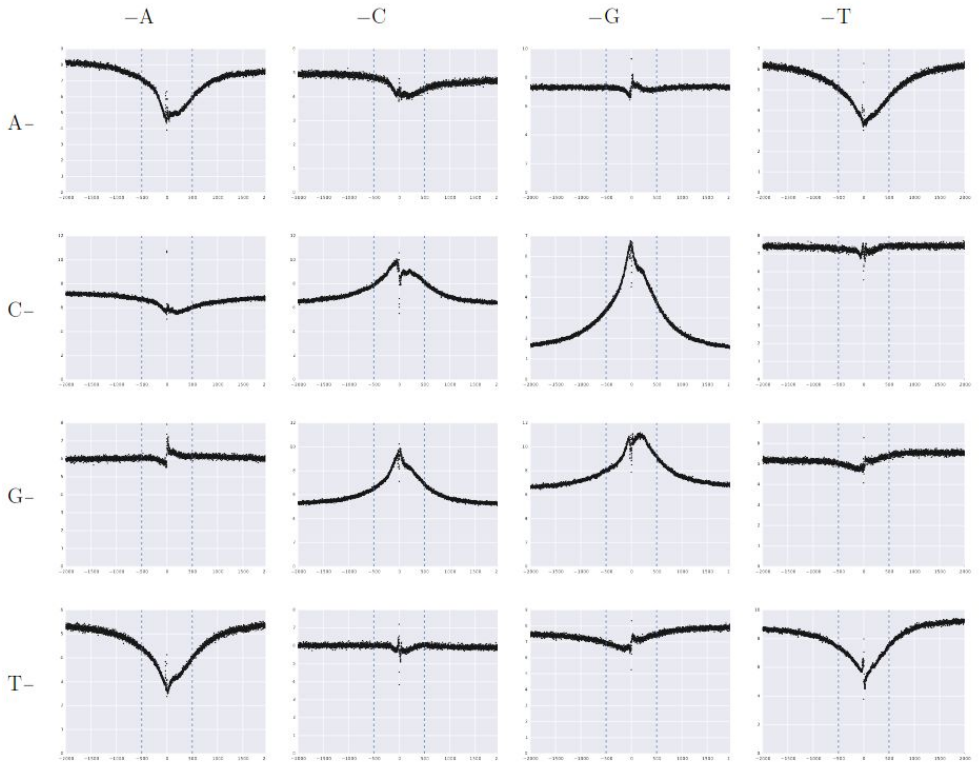


Fig S2

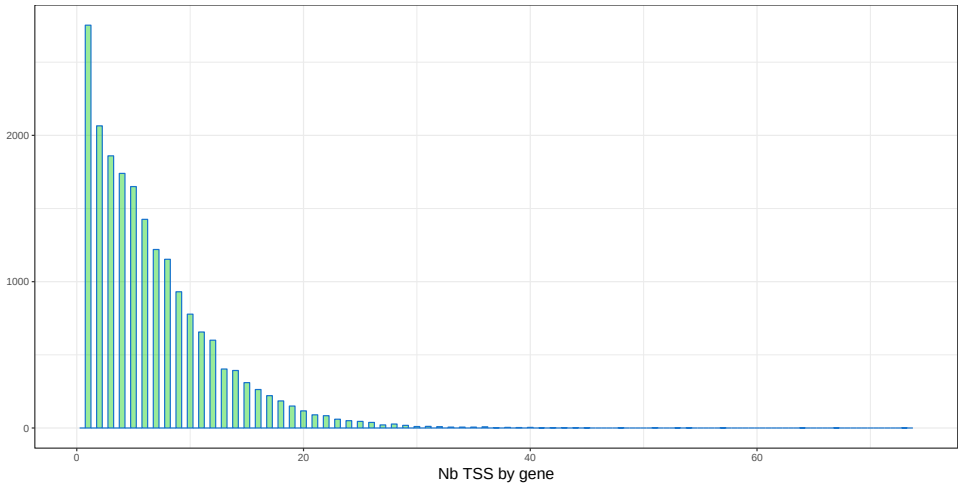


Fig S3

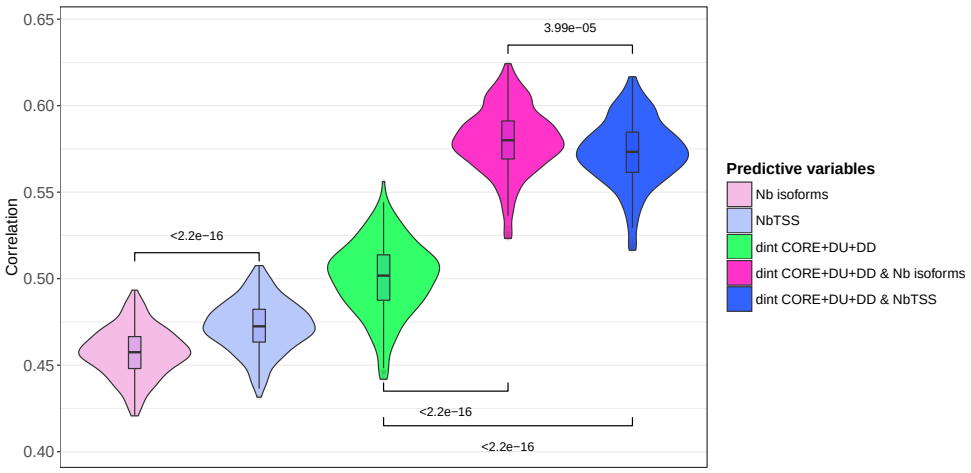


Fig S4

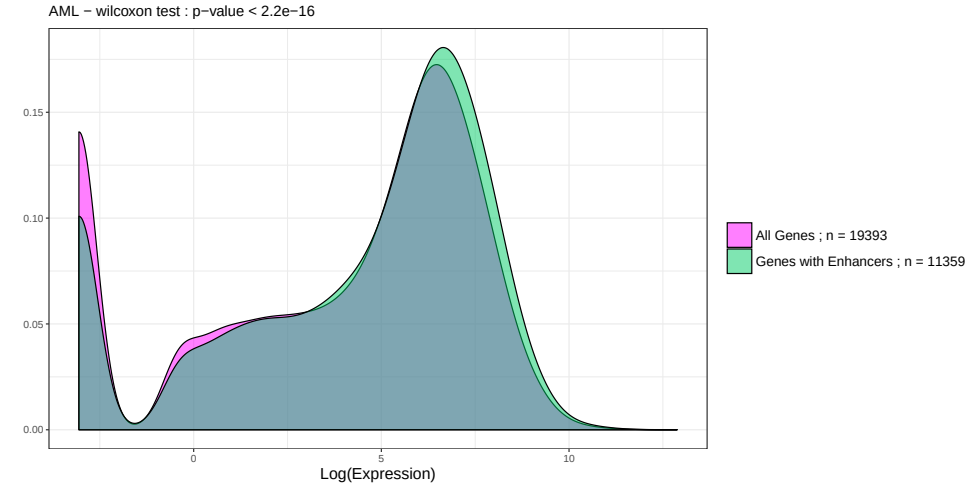
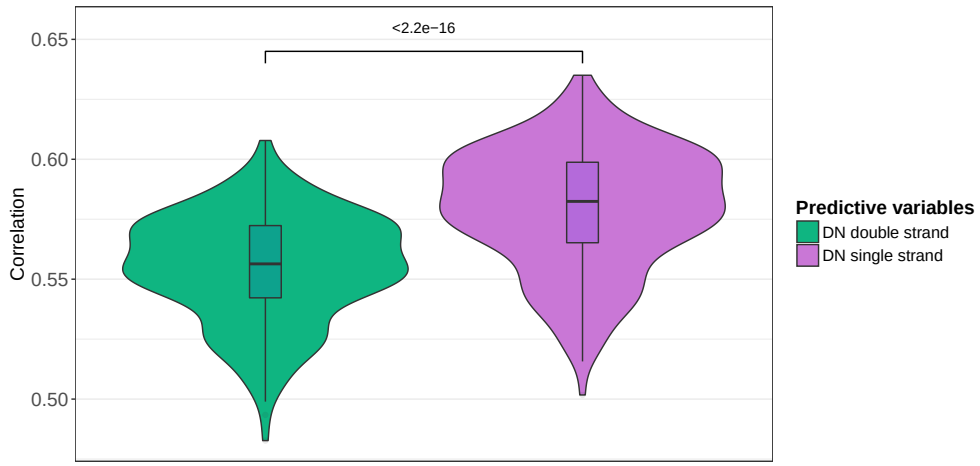


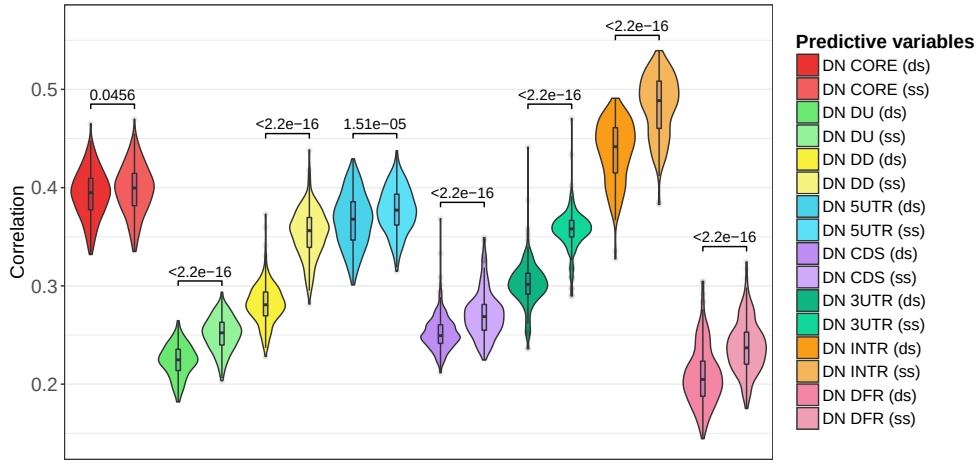
Fig S5

	INTR	5UTR	3UTR	CDS	CORE	DD	DFR	DU
STEP 1	0.4885	0.3771	0.358	0.2688	0.3996	0.3562	0.2369	0.2279
STEP 2		0.5298	0.5242	0.5069	0.5211	0.5037	0.4929	0.4887
STEP 3			0.5517	0.5488	0.5397	0.5391	0.5368	0.5306
STEP 4				0.5657	0.5587	0.5583	0.5575	0.553
STEP 5					0.5728	0.5718	0.5693	0.567
STEP 6						0.5781	0.5779	0.5733
STEP 7							0.5813	0.5786
STEP 8								0.5824

Table S2

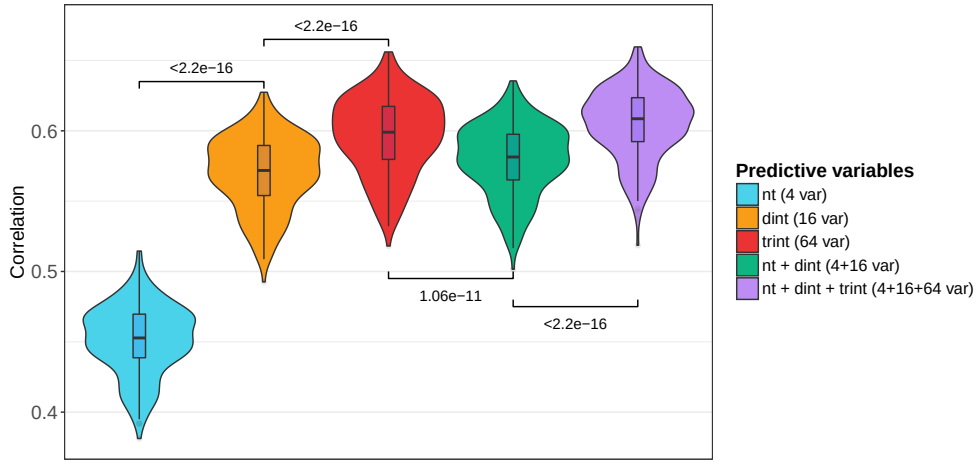

0.65
0.60
0.55
0.50
Correlation
<2.2e−16
Predictive variables
DN double strand
DN single strand

A

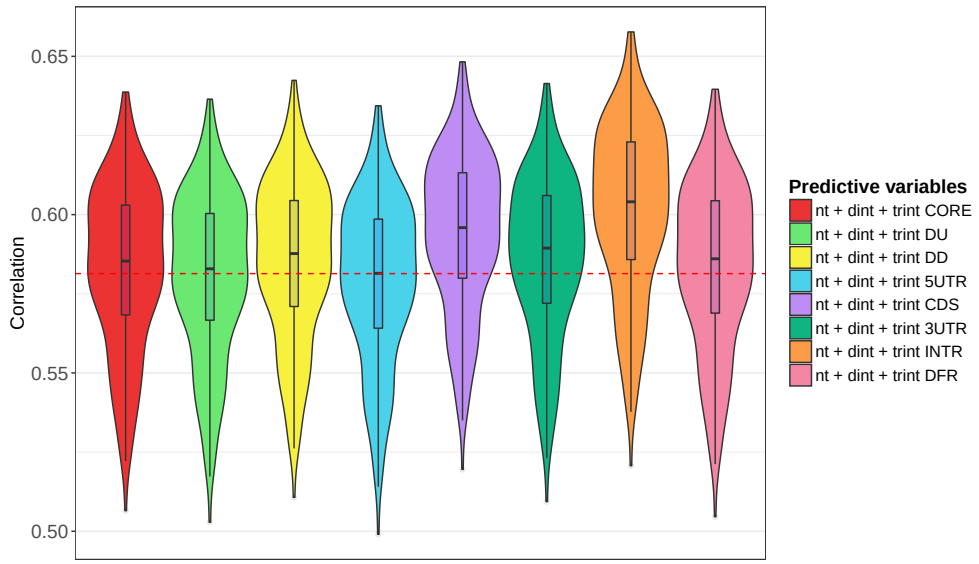

0.5
0.4
0.3
0.2
Correlation
0.0456
<2.2e−16
<2.2e−16
1.51e−05
<2.2e−16
<2.2e−16
<2.2e−16
<2.2e−16
Predictive variables
DN CORE (ds)
DN CORE (ss)
DN DU (ds)
DN DU (ss)
DN DD (ds)
DN DD (ss)
DN 5UTR (ds)
DN 5UTR (ss)
DN CDS (ds)
DN CDS (ss)
DN 3UTR (ds)
DN 3UTR (ss)
DN INTR (ds)
DN INTR (ss)
DN DFR (ds)
DN DFR (ss)

B

Fig S6


<2.2e−16
<2.2e−16
1.06e−11
<2.2e−16
Correlation
0.6
0.5
0.4
Predictive variables
nt (4 var)
dint (16 var)
trint (64 var)
nt + dint (4+16 var)
nt + dint + trint (4+16+64 var)

A


Correlation
0.65
0.60
0.55
0.50
Predictive variables
nt + dint + trint CORE
nt + dint + trint DU
nt + dint + trint DD
nt + dint + trint 5UTR
nt + dint + trint CDS
nt + dint + trint 3UTR
nt + dint + trint INTR
nt + dint + trint DFR

B

Fig S7

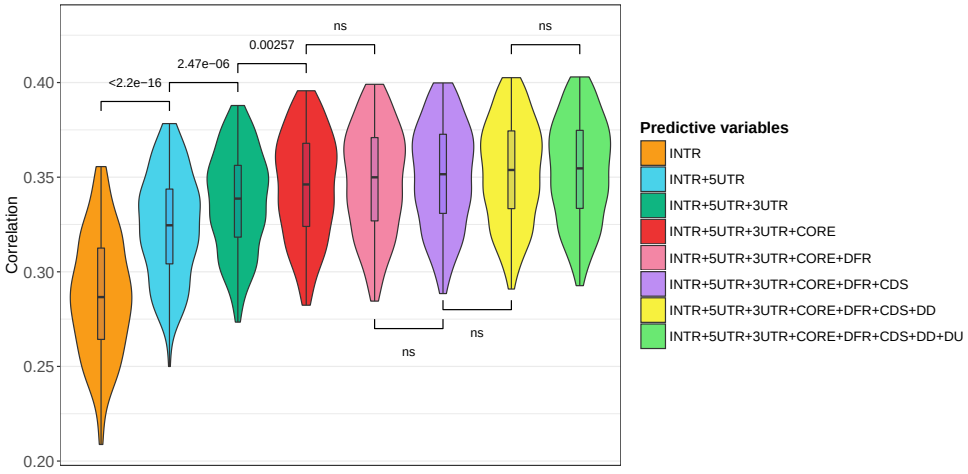


Fig S8

	INTR	5UTR	3UTR	CORE	DFR	CDS	DD	DU
STEP 1	0.2866	0.2462	0.1879	0.2417	0.1577	0.1392	0.1757	0.1358
STEP 2		0.3246	0.3080	0.3182	0.2918	0.2959	0.2931	0.2898
STEP 3			0.3387	0.3348	0.3275	0.3286	0.3276	0.3248
STEP 4				0.3462	0.3416	0.3429	0.3416	0.3396
STEP 5					0.3499	0.3487	0.3487	0.3458
STEP 6						0.3516	0.3500	0.3478
STEP 7							0.3538	0.3515
STEP 8								0.3547

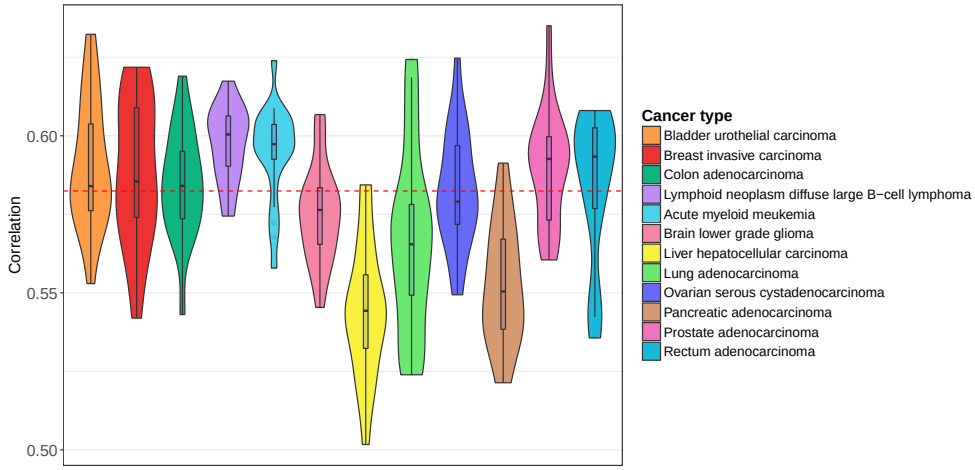
Table S3

Gene ontology term	Count	Benjamini corrected P-value
Cellular macromolecule metabolic process	612	1.8E-23
Cellular metabolic process	681	1.2E-16
Cellular protein metabolic process	390	2.8E-16
Macromolecule metabolic process	624	4.0E-16
Nucleic acid metabolic process	404	4.0E-16

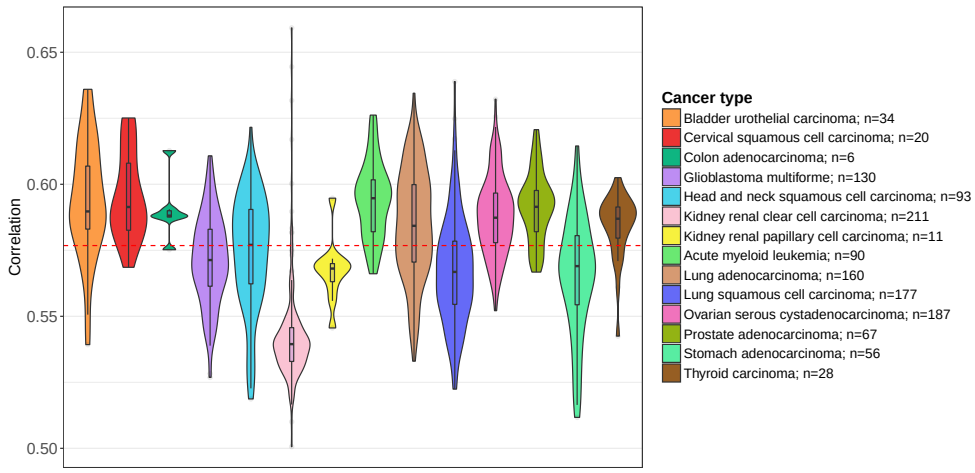
Table S4

Gene ontology term	Count	Benjamini corrected P-value
Positive regulation of cellular process	528	7.0E-14
Nervous system development	284	1.3E-13
Positive regulation of macromolecule metabolic process	346	3.5E-12
Positive regulation of biological process	565	8.1E-12
Neurogenesis	200	5.9E-11

Table S5



A



B

Fig S9

	CORE	5UTR	CDS	3UTR	INTR	DFR
eQTL density	$5.29 * 10^{-4}$	$5.54 * 10^{-4}$	$1.28 * 10^{-4}$	$1.78*^{-4}$	$7.03 * 10^{-5}$	$1.49 * 10^{-4}$

Table S6

SNP density	CORE	INTR
chr1	0.03937541	0.02680815
chr19	0.05870189	0.04408599
chr2	0.06016752	0.04202322

Table S7

	5UTR	CDS	3UTR	INTR	DFR
enh density	$8.27 * 10^{-4}$	$2.44 * 10^{-5}$	$6.41 * 10^{-4}$	$5.14*^{-3}$	$4.08 * 10^{-3}$

Table S8

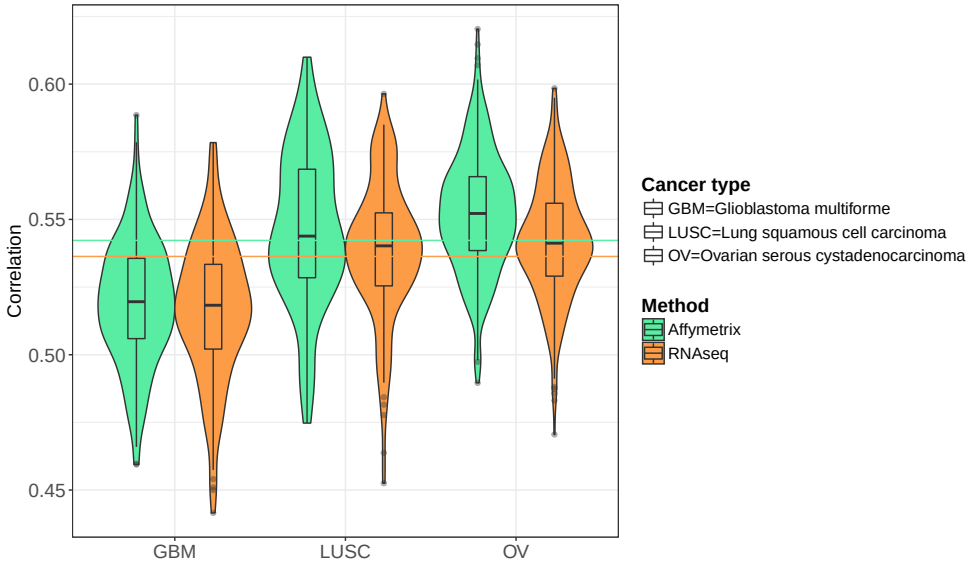


Fig S10

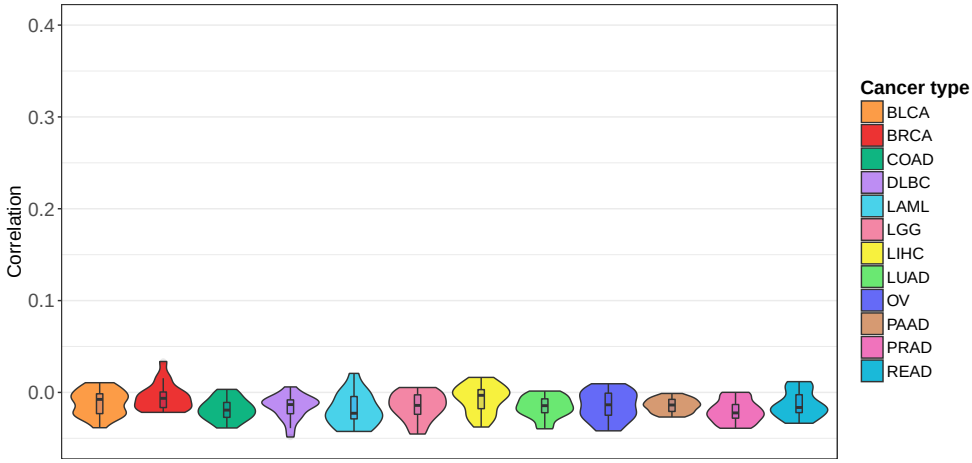


Fig S11

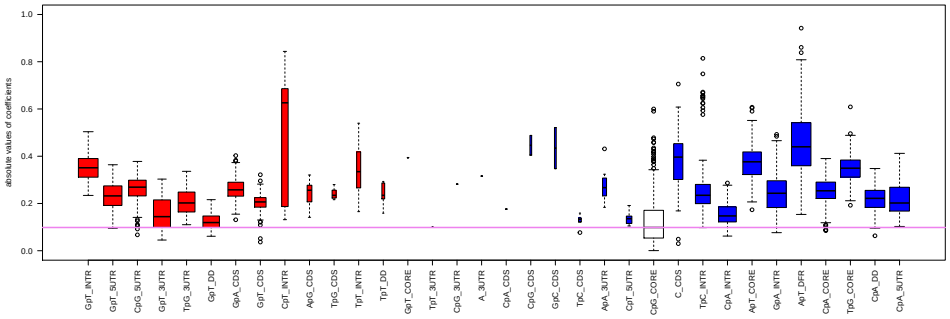
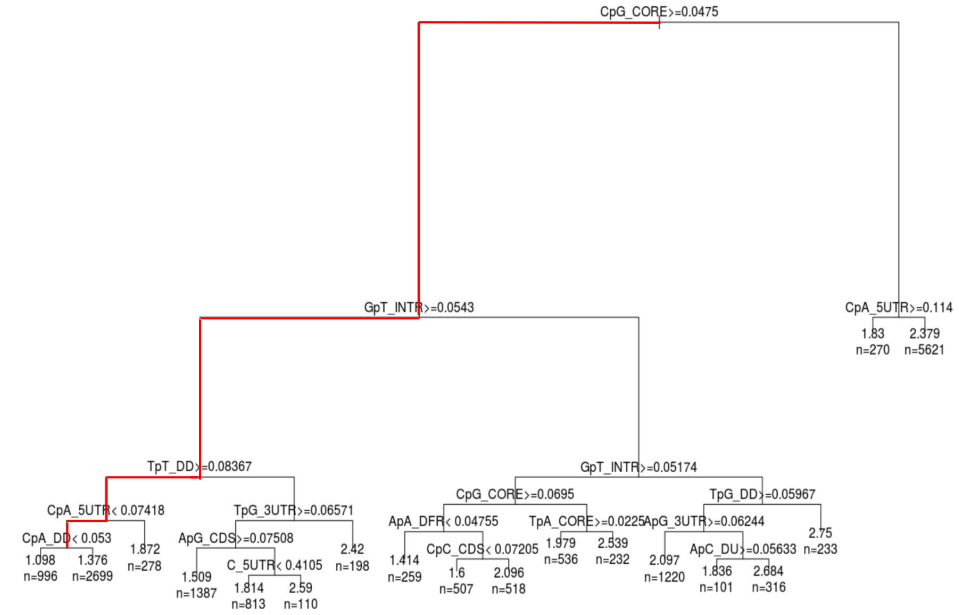
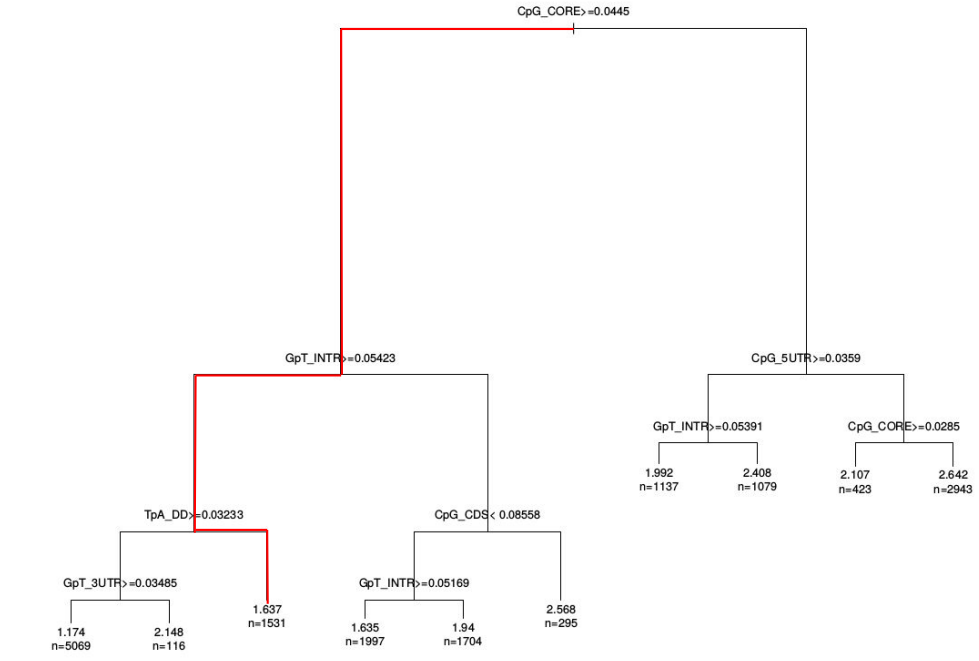


Fig S12


CpG_CORE>=0.0475
GpT_INTR>=0.0543
CpA_5UTR>=0.114
1.83
n=270
2.379
n=5621
TpT_DD>=0.08367
GpT_INTR>=0.05174
CpA_5UTR< 0.07418
TpG_3UTR>=0.06571
CpG_CORE>=0.0695
TpG_DD>=0.05967
CpA_DD< 0.053
1.872
n=278
ApG_CDS>=0.07508
2.42
n=198
ApA_DFR< 0.04755
TpA_CORE>=0.0225
ApG_3UTR>=0.06244
2.75
n=233
1.098
n=996
1.376
n=2699
1.509
n=1387
C_5UTR< 0.4105
2.59
n=110
1.414
n=259
CpC_CDS< 0.07205
1.979
n=536
2.539
n=232
2.097
n=1220
ApC_DU>=0.05633
1.814
n=813
1.6
n=507
2.096
n=518
1.836
n=101
2.684
n=316

A


CpG_CORE>=0.0445
GpT_INTR>=0.05423
CpG_5UTR>=0.0359
GpT_INTR>=0.05391
CpG_CORE>=0.0285
1.992
n=1137
2.408
n=1079
2.107
n=423
2.642
n=2943
TpA_DD>=0.03233
CpG_CDS< 0.08558
GpT_3UTR>=0.03485
1.637
n=1531
GpT_INTR>=0.05169
2.568
n=295
1.174
n=5069
2.148
n=116
1.635
n=1997
1.94
n=1704

B

Fig S13

Chapter 3

Attempt to improve model performances

In the first chapter, we fitted gene expression in function of nucleotide compositions, and we focused on the biological significance of the selected nucleotide compositions. A penalized linear model followed by a stability selection method were proposed for this aim. Although these model performances are satisfying compared to a model fitted with experimental data, one can think about different approaches to increase these performances. In this chapter, the primary objective is model accuracy and model prediction improvement. The model we attempt to improve is the one presented in the previous Chapter with the highest performances (Chapter 2, Figure 4) *i.e* the Lasso penalized linear regression fitted on 16,294 genes, to predict gene expression using 160 nucleotide compositions (20 nucleotide composition in each of the 8 regulatory regions (Chapter 2, Figure 1)) .

On one hand, a way to improve model performances is to add to the model variables that are biologically significant to predict gene expression. It is known that gene regulation is related to different regulatory regions of DNA and that these regions interfere together to transcribe the gene. Hence, we propose including interactions between nucleotide compositions in different regions to predict gene expression with higher accuracy.

On another hand, low model performances may be related to specific model hypotheses

that may not be satisfied. The model that we considered in the previous chapter is based on a linear relationship between the gene expression (log transformed) and the nucleotide compositions. This hypothesis might not be valid and limits its performances. To overcome this limit, one can think about variable transformations that allow a non-linear relationship between gene expression and nucleotide compositions.

Different approaches in the literature allow taking into account variable interactions and non-linear relationships. First, in chapter 2, we used Random Forests [Bre01]. These are non-linear models where, at each node, a threshold is selected, and the continuous variable is transformed into a binary decision variable based on a threshold. The performances of this method for predicting gene expression from the nucleotide compositions of the eight chosen DNA regions were similar to those of the Lasso penalized linear regression (Chapter 2, Table S1). However, defining a transformation rule based on Random Forests is not straightforward given that, for each tree, the variables and the binary decisions are different. Although, Random Forest were developed and used to detect interactions [BKBY18] (see Section 1.4.4). This method was presented in a classification framework and generalized to a regression framework, but the application of this algorithm in regression requires a transformation of the continuous responses into a range of interests (classes) using clustering methods [BKBY18]. However, MARS model [Fri91] was built with the same aim as our approach, *i.e.* fitting a regression model with non-linear transformed variables and interactions. This last one is used for comparison with our approaches.

In this chapter, we study each case (interactions and transformations) using different models and algorithms. First, we define a sampling method for evaluation in Section 3.1. In Section 3.2, we consider adding variable interactions to the original model. Then, a non-linear transformation framework is presented in Section 3.3. In this Section, new variables are built using different forms of known transformations then a penalized model is fitted with these variables. Finally in Section 3.4 we fit a model based on both approaches *i.e* a model with transformed variables as well as interactions between these variables and compare our results with the MARS model.

3.1 Sampling

In the analysis performed in this chapter, two types of sampling are used, one on the individuals and one on the conditions (tumors) whose number is vast in the database TCGA.

3.1.1 Individuals for training and validation

The studies presented in this chapter require to train, validate and select different models and variables. In order to avoid over-fitting, we randomly partition the data into three datasets (in total we have 16294 genes). One dataset ***(test subset)*** of 2000 genes is used only to evaluate the performance of the penalized linear model (Spearman correlations). The rest of the genes are randomly separated in two datasets of equal size, one ***(training 1 subset)*** is used to build novel predictive variables (non-linear transformations or combinations of variables, ...) and the other ***(training 2 subset)*** is used to fit a Lasso penalized linear regression.

3.1.2 Number of conditions

First, gene expression in different tumors of same cancer has very similar distributions. Figure 3.1 shows the log of the gene expression in 6 tumors, two for each type of cancer: i) Ovarian (OV), ii) Acute Myeloid Leukemia (AML) and iii) Prostate Adenocarcinoma (PRAD). We notice that in each type of cancer, the distributions, as well as the median expression, are similar. However, these distributions are distinct from one type of cancer to another. For this reason, and considering that this is an exploratory phase, we decide to restrict this study to 12 tumors, one for each type of cancer considered in Chapter 2 defining our reference model.

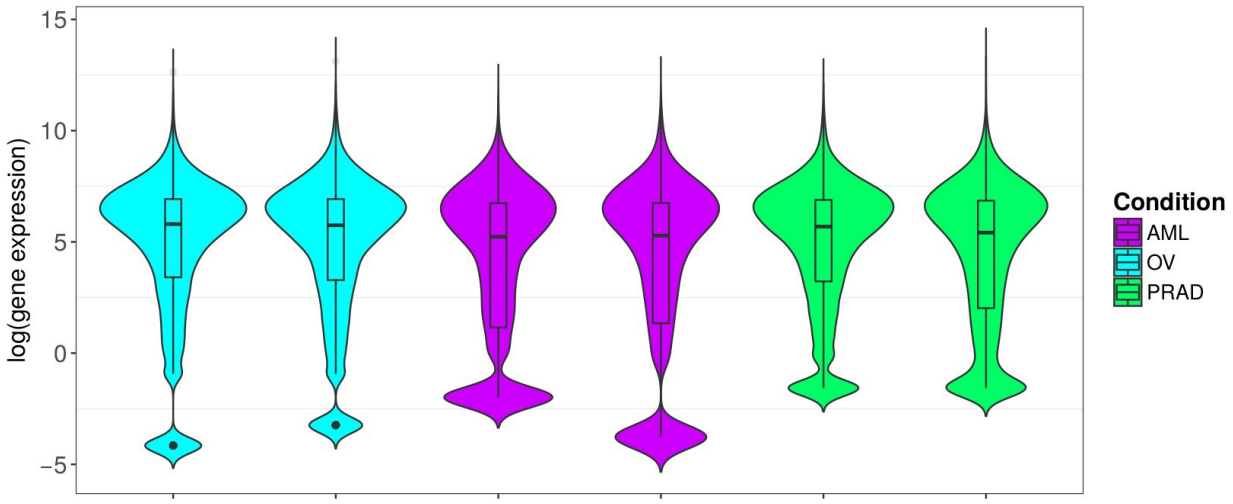


Figure 3.1: ***Distribution of gene expression in tumors from three different types of cancer****. For each type of cancer: i) Ovarian (OV) in cyan, ii) Acute Myeloid Leukemia (AML) in purple and iii) Prostate Adenocarcinoma (PRAD) in green, the distribution of gene expression in two tumors is represented. Each boxplot contains 16294 values one for each gene.*

3.2 Contribution of second-order interactions between nucleotide compositions

In Section 1.4, we presented different papers that studied interactions between DNA regions as well as their influence on gene regulation ([TOHTS13], [RSC+15], [SYPM16]). In general, interactions may appear between two or more components, but in this thesis, only second-order interactions are discussed. Higher order interactions are not tested for computational reasons knowing that this is an exploratory phase. We consider all pairs of nucleotides (and di-nucleotides) interactions between two regulatory regions (inter regions interactions) and within one regulatory region (intra region interactions). In total for 160 nucleotide compositions, the number of second order interactions is a combination without repetition of 2 among 160 which results in 12720 second-order interactions.

Our objective is to fit an ℓ_1 penalized linear model (Lasso) based on nucleotide compositions and their second-order interactions, to explain and predict gene expression. One can think about fitting the model using all possible second-order interactions (12720 variables) and let the ℓ_1-regularization procedure selects essential interactions. Two possible aspects

of these variables may limit the model and bias the results: i) a high number of variables, among which only a few variables are associated with gene expression, and ii) high correlations between interaction variables. To avoid this problem, in a first step, we set a selection algorithm to pre-select nucleotide interactions with a considerable influence on gene expression. The selected interactions are added to simple nucleotide compositions (noted by the following original variables) to fit a Lasso penalized linear regression in a second step. The performances of the model fitted in step 2 (original and interactions variables) are compared to those of the model fitted with only the original variables. Furthermore, we proceed with a stability selection algorithm (see Section 1.2.1.3) to select interactions that are stable and relevant to the model.

3.2.1 Algorithm for pre-selecting interactions

As we presented in Section 1.4, different approaches exist to detect interactions. Our work was inspired by the approach presented by of Wein-Yin Loh (2002) [Loh02] based on χ^2 test. In this section, we use the χ^2 test of independence to pre-select second-order interactions based on their influence of gene expression. These variables are then used in a penalized linear model fitted to predict gene expression. First, we should define the type of interactions that we consider. Let X_i be the vector of the i^{th} nucleotide composition, X_j be the vector of the j^{th} nucleotide composition and Z_{ij} the interaction between X_i and X_j. We define three types of interactions: i) Conjunction (and): $Z_{ij} = \text{minimum}(X_i,X_j)$. ii) Disjunction (or): $Z_{ij} = \text{maximum}(X_i,X_j)$. iii) Product: $Z_{ij} = \text{product}(X_i,X_j)$. The idea of the product form of interaction comes from the linear model that define interactions as the product of two variables. Algorithm 4 describes our two-stage procedure for selecting interactions using the $p-values$ of different χ^2 tests. In the first stage, we set a selection rule by a threshold defined on the χ^2 test $p-values$ computed on the original variables (step 2 in Algorithm 4). The second stage is based on computing χ^2 test $p-values$ for all interaction variables and pre-select those that verify the selection rule (step 2 in Algorithm 4). For this Section, the selection rule is defined as: an interaction is selected if its adjusted p-value ($P^{adj}_{interaction}$: Algorithm 4 step 2 (b)) is lower than the minimum of the adjusted p-values of the χ^2 test computed with the original variables (*i.e.* $P^{inf}_{original}$ Algorithm 4 step

1 (b)). The algorithm is applied three times for each tumor on the *training 1 subset*, one for each defined form of interaction (minimum, maximum and product).

Algorithm 4: χ^2 test of independence to pre-select second order interactions

Calculate the residuals from a constant model (mean) fitted to gene expression data. Each gene is then classified as positive or negative whether if its residual is positive or negative respectively.

Step 1 - Selection threshold definition:

a) For each variable X_i, divide the data into four groups at the sample quartiles. Construct a 2×4 contingency table with the signs of the residuals (positive versus negative) as rows and the groups as columns then count the number of observations in each cell of the table. Compute a χ^2-statistic and its p-value from a χ^2_3 distribution. This test is noted as **original test**.

b) Compute adjusted p-values for the **original tests** (160 values) using the false discovery rate (FDR) method [BH95]. Let $P^{inf}_{original}$ be the smallest adjusted p-value of the **original tests**.

Step 2 - Interaction selection:

a) For each defined interaction Z_{ij}, divide the data into four groups at the sample quartiles. Construct a 2×4 contingency table with the signs of the residuals as rows and the groups as columns then count the number of observations in each cell of the table. Compute a χ^2-statistic and its p-value from a χ^2_3 distribution. This is noted for next as **interaction test**.

b) Compute $P^{adj}_{interaction}$ adjusted p-values for the **interaction tests** (12720 values) using FDR. The interactions are pre-selected with respect to a selection rule defined base on the threshold in step 1.

The average number of pre-selected interactions by tumors is 5, 1, and 3 respectively for the minimum, maximum and product definition. In Figure 3.2 are presented the pre-selected second-order interactions over the 12 tumors for the minimum and the product operator

(names at each barplot). Besides, for each interaction, the bar represents the number of tumors in which it is selected. For example in (a), when using the minimum operator, the interaction between the CpG in the CORE promoter (CpG_CORE) and the GpT in the introns (GpT_INTR) is pre-selected in all tumors (12) while the intersection between theCpG in the CORE promoter (CpG_CORE) and the GpA in the CDS region (GpA_CDS) was pre-selected only in one tumor (of Type BRCA). When using the maximum operator, just the interaction between CpG in the CORE promoter (CpG_CORE) and CpG in 5'UTR (CpG_5UTR) is pre-selected in each of the 12 tumors.

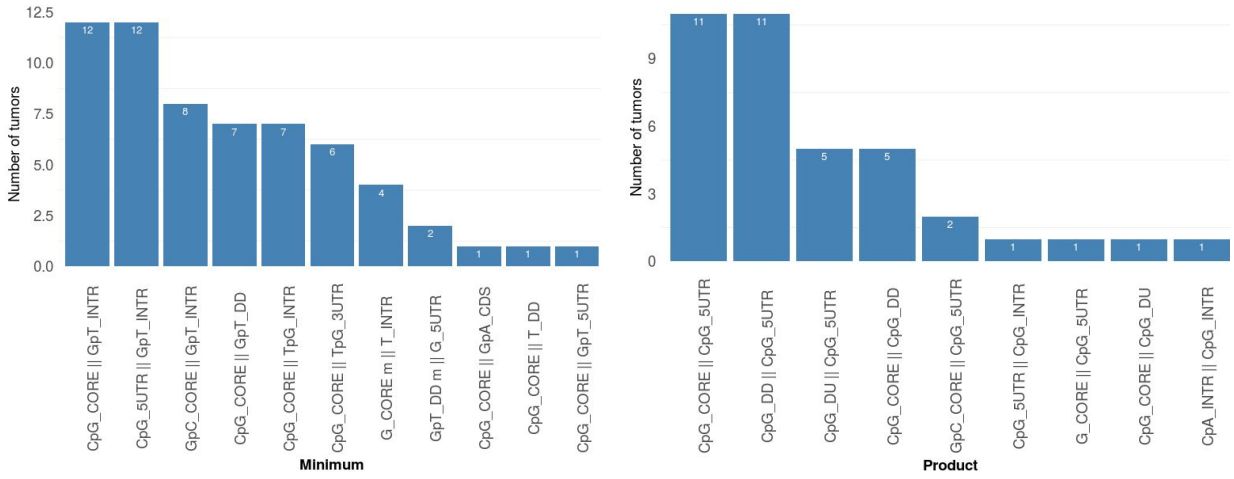


Figure 3.2: ***Nucleotide interactions pre-selected by our procedure described in Algorithm 1.*** *In each plot, each bar represents the number of tumor in which the correspondent interaction (i.e names on x-lable) is pre-selected by our procedure. a) shows pre-selected interactions using the minimum operator while b) shows pre-selected interactions using the product operator.*

These pre-selected interactions in each tumor, are then considered in addition to original variables (160 nucleotide composition) as predictive variables in a ℓ_1 penalized linear model fitted to predict gene expression.

3.2.2 Interaction operators evaluation and comparison

The first step of the method was to pre-select interactions that may be related to gene expression. The next step is to compute the contribution of these novel variables in a

Lasso penalized linear regression fitted to predict gene expression. As mentioned in Section 3.2.1, we tested three definitions of interactions: the minimum, the maximum and the product of pairs of nucleotide composition, both intra and inter regions. First, we compare the contribution of each operator separately, and then we test different combinations of operators.

First, we evaluate the contribution, in term of increasing model performances, for each type of operators. For that, we fit three Lasso penalized linear regressions on three sets of variables, each using original variables and pre-selected interactions for each operator independently. For each set of variables, models are fitted for 12 tumors on the *training 2 subset* and evaluated on the *test subset*. Spearman correlations are presented in Figure 3.3 (boxplots in green). The three different operators show a small increase in model performances with median Spearman correlations respectively of 0.5654, 0.5624, and 0.5622 for the minimum, product, and maximum operators compared to a median correlation of 0.5609 for the reference model (with original variables). Furthermore, the interactions defined by the minimum between two variables showed the highest increase when added to the model.

Secondly, we consider combining the pre-selected interactions between two or three different operators. Lasso penalized regression models were fitted for three different sets of predictive variables. Just like before, for each combination, models are fitted for 12 tumors on the *training 2 subset* and evaluated on the *test subset*. The different sets contain original variables (160 nucleotide compositions) then we add: i) pre-selected interactions defined by the minimum and the product operator, ii) pre-selected interactions defined by the minimum and the maximum operator, and iii) pre-selected interactions defined by the maximum and the product operator. The final considered Lasso penalized regression model is fitted using original variables as well as pre-selected interactions defined by the three different operators.

Spearman correlations of the three sets of predictive variables (boxplots in blue) as well as that of the model with all variables (boxplot in cyan) are presented in Figure 3.3 with respective median Spearman correlations of i) 0.5654, ii) 0.5650, iii) 0.5625 and 0.5656 for the complete model.

The performances were slightly similar, with a maximum increase of 0.47% when adding all the pre-selected interactions. This result may be explained by the few numbers of interactions that are pre-selected using the selection rule in Algorithm 4. One can think of changing the selection rule to increase the number of pre-selected interactions. This is performed in the next subsection.

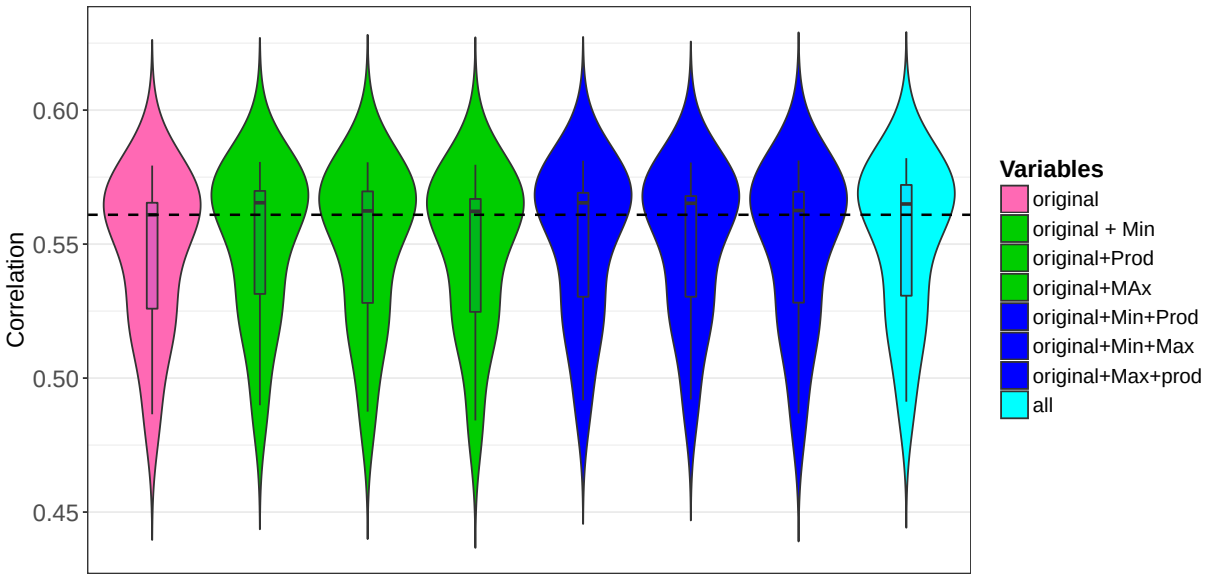


Figure 3.3: ***Comparing the performances for different interaction combinations*** *The first boxplot (pink) refers to the reference model fitted with original variables. Boxplots in green are for models fitted with 2 types of variables: original and selected interactions defined by (from left to right) minimum, product and maximum. Blue boxplots are for models fitted with original variables in addition to the set of selected interactions defined by two different operators, from left to right: minimum & product, minimum & maximum and maximum & product. The last boxplot (cyan) represents a model fitted with original variables and all the interactions selected for the 3 operators. Note that Spearman correlations for different models are calculated for 12 tumors using the test subset. The horizontal black line represents median correlation of the reference model fitted with only original variables.*

3.2.3 Changing the selection rule using minimum interactions

In Section 3.2.1, we presented an algorithm to select interactions with a strong selecting rule where an interaction is selected if only its χ^2 test p-value ($P^{adj}_{interaction}$) is lower than the threshold ($P^{inf}_{original}$) *i.e.* computed based on the χ^2 independence test for the original variables. Model performances do not increase significantly, and one of the causes may be the low number of pre-selected interactions. We propose in this Section to resolve this by changing the selecting rule to increase the number of pre-selected interactions. For computational reasons, we only focus on interactions defined by the minimum operator, the one that had, when tested separately, the higher contribution on model performances in term of correlation between observed and predicted gene expression on the *test subset* (see Figure 3.3).

To increase the number of pre-selected interactions progressively, we increase the threshold of the selecting rule successively. Instead of setting the threshold to the lowest p-values of original test ($P^{inf}_{original}$), we set three different thresholds to test: i) $P^{q2}_{original}$: the quantile 2%, ii) $P^{q5}_{original}$: the quantile 5% and iii) $P^{q10}_{original}$: the quantile 10%, of the adjusted p-values calculated from tests for original variables.
When increasing the threshold of the selection rule, the number of selected interactions increase as well. In the first column of Table 3.1 figure the mean numbers of pre-selected interactions using each one of the four different thresholds. We notice that the number of pre-selected interactions is exponential in function of the threshold.

In order to select a threshold (*i.e* the number of pre-selected interactions), we fit Lasso penalized regression models on 4 different sets of variables, one for each threshold, on the *training 2 subset* using original variables and pre-selected interactions corresponding to each threshold. Median Spearman correlations, over 12 tumors, between predicted and observed gene expression, calculated on the *test subset*, are presented in the second column of Table 3.1.

From Table 3.1, we notice that model performance increases slightly with the increase

threshold	mean number of interactions	median Spearman correlation
$P^{inf}_{original}$	5	0.565
$P^{q2}_{original}$	150	0.567
$P^{q5}_{original}$	366	0.569
$P^{q10}_{original}$	801	0.572

Table 3.1: ***Summary of numbers of pre-selected interaction and Spearman correlations for each used threshold.*** *$P^{inf}_{original}$, $P^{q2}_{original}$, $P^{q5}_{original}$ and $P^{q10}_{original}$ refers respectively to the quantiles of order 0% (inf), 2%, 5% and 10% of the adjusted p-value calculated from a χ^2 test on the original variables. The means of number of selected interactions and the medians of Spearman correlation are calculated over 12 tumors. Correlations are calculated on the* test subset.

in the number of pre-selected interactions, but the improvement is minimal compared to the number of variables added to the model. Furthermore, with the aim to study the stability of each interaction and select interactions that are significant in the Lasso penalized regression model, we consider the model using the 2% quantile threshold with a mean number of interactions of 143 and a median Spearman correlation of 0.567.

3.2.4 Variable stability and region interactions selection

To go further in this procedure and study the explanatory power of the interaction variables, we consider the threshold defined by the 2% quantile, leading to the selection of 143 interactions in average (see the previous Section). Using these interactions as well as the original variables we run a stability selection algorithm (1.2.1.3) and select the stable variables. For this study, we consider a variable as stable if its selection frequency in the model over 500 repetitions is higher than 70% (the choice of parameters is explained in 1.2.1.3). Table 3.2 shows the number of stable variable in each tumor when using original variables (first row), and when using original variables and pre-selected interactions (second row) as well as the ratio of the number of stable interactions, with respect to the total number of stable variables (last row) of the model.

	tum1	tum2	tum3	tum4	tum5	tum6	tum7	tum8	tum9	tum10	tum11	tum12	mean
Original variables	15	16	16	14	15	17	14	17	16	17	15	16	16
Orig + interactions	20	19	20	19	19	22	22	19	19	21	19	18	20
% of interactions	65%	58%	65%	53%	53%	41%	64%	58%	53%	57%	63%	50%	56%

Table 3.2: ***Number of stable variables with interactions.*** *For each tumor, first and second row represent the number of stable variables in: i) model with only original variables. ii) model with original variable and pre-selected interactions. The third row is the percentage of stable interactions among stable variables from the model in second row. The last column is the mean of each row.*

We can see that the mean number of stable variables when using both original variables and interactions is slightly higher than that when using only original variables (20 Vs. 16). More importantly, more than half of the stable variables are pre-selected interactions. To compute the contribution of only stable variables in predicting gene expression, we run a linear regression model (without regularization) to predict gene expression in each tumor using only stable variables in both cases (with or without interactions). The median Spearman correlations between observed and predicted gene expression are 47% and 52% respectively for the model fitted with only original variables and the model fitted with original variables and interactions. Hence, from an explicative point of view (using stable variables only), adding interactions of nucleotide compositions to the model brings a substantial gain (plus 5% correlation between observations and predictions).

We further study the importance of interactions when the original variables are excluded from the model. First, the median Spearman correlation of a Lasso penalized regression model fitted with only pre-selected interactions is 0.565. The performances are similar to those of a model fitted with both original variables and interactions with a median correlation of 0.567. However, when running a stability selection algorithm for a regularized model using only interaction variables, only two interactions are stable in each tumor. This result may be explained by the fact that only interactions are not sufficient to predict gene expression. Further, the interactions selected by a Lasso penalized regression model

reflect somehow the information of original variables. The non-stability of the variables may be explained by the fact that at each step, to compensate for the information given by an original variable, the Lasso regularization procedure selects one of the numerous interactions with this variable. For example, if CpG_CORE is necessary for the model, the Lasso penalized regression model can select any interaction that contains CpG_CORE and not necessarily the same at each step.

3.3 Introducing non-linear transformations of nucleotide compositions

In this section, we aim to consider non-linear relationships between gene expression and nucleotide composition. This is already partially addressed in Chapter 2 with regression trees and random forests. The performances are similar to those of ℓ_1 penalized linear model but studying each variable, and the explicit form of its non-linearity with the response variable is not straightforward. In this section, we provide a framework to evaluate more general non-linear relationships between gene expression and nucleotide composition in two consecutive steps. The first step consists in finding appropriate non-linear transformations of the variables and the second step, in including transformed variables in a Lasso penalized linear regression to evaluate their importance. Each step is applied to a different subset to avoid overfitting. In the following, we present different methods to adjust non-linear transformations. The results of each transformation are compared to the model with the original variables (see Chapter 2, Figure 4). Finally, the best model of each transformation are compared (with MARS) in Section 1.2.3.3

3.3.1 Basic transformations

When we talk about non-linearity, one would at first think about testing basic form of non-linear mathematical transformations. In this part, we consider four different forms of non-linearity: square, cube, exponential and logarithmic transformations. Note that for logarithmic transformation, zeros are replaced by 10^{-5}. In this section, we do not fit a model to build our transformation model. Hence this method do not require to be applied

on the *training 1 subset* first. However, to compare this transformation with other types of transformations that require a sampling, we build transformations directly on nucleotide composition from the *training 2 subset*.

For each one of these non-linear transformations (square, cube, exponential and logarithmic), we fit two ℓ_1 penalized regression models on the *training 2 subset* i) one using only transformed variables (160 variables) and ii) one using both original and transformed variables (320 variables). These models are then evaluated by Spearman correlation between predicted and observed gene expression on 2000 genes from the (*test subset*). Figure 3.4 illustrates boxplots of these correlations computed in 12 tumors for the various fitted models. Notice that each one of these non-linear transformations increases the model performance. Compared to the model fitted with only original variables (median Spearman correlation = 0.5609), models fitted with only **square** transformed variables present the highest increase (median Spearman correlation = 0.578). When adding original variables to the transformed ones, square and exponential transformations show similar performances (median Spearman correlation = 0.583).

Elementary non-linear transformations of nucleotide compositions have shown significant improvement in explaining gene expression. This gives credit for a non-linear relationship. Thus we pursue further with more complex non-linear transformations. Next, we present three non-linear transformation models: loess regression, hockey stick regression, and piecewise regression. Finally, these models are compared to MARS.

Algorithm 5 explains briefly the steps to be followed for each transformation model to build the new variables and uses them to predict gene expression.

3.3.2 Loess regression

Loess regression is a non-linear regression model based on the k-nearest neighbors (see Section 1.2.2.3). As already presented, fitting a loess regression requires initialization of two parameters: i) the degree of the polynomial that can be 0, 1 or 2 and ii) α the percentage of k-neighbors. In general, α is known as the span and can take values between 0 and 1 (including). Usually, three values are used: 25%, 50%, and 75%. The loess transformations

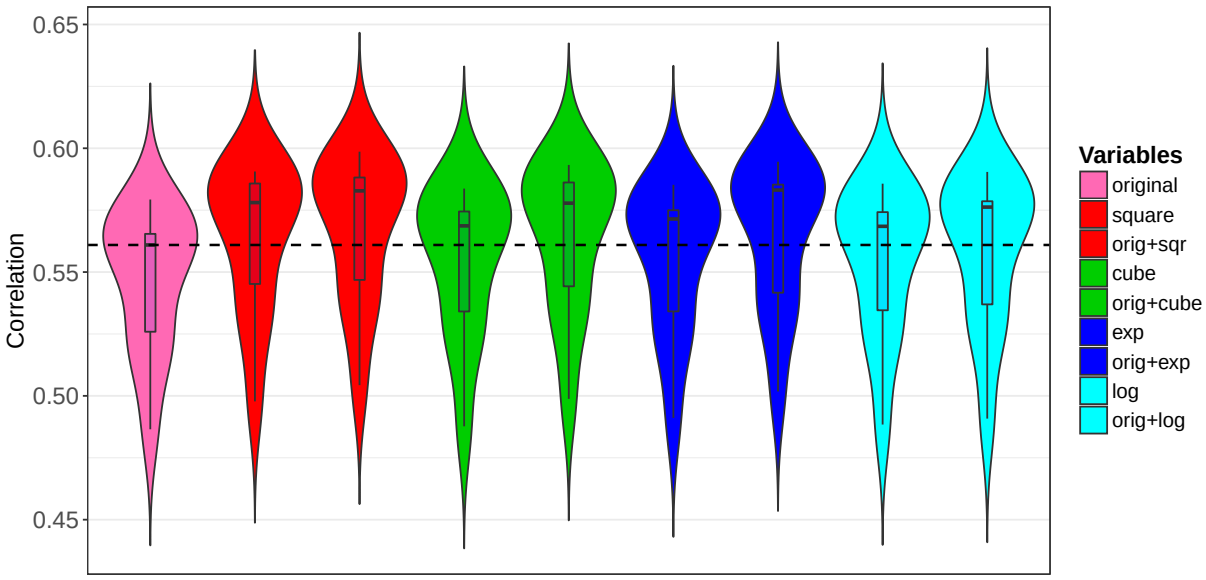


Figure 3.4: ***Basic non-linear transformations*** *The first boxplot in pink represents the correlation coefficients using variables in their original form. For each type of transformations, two boxplots of correlations are shown: one for a model using only transformed variables (first boxplot), and one using original and transformed variables (second boxplot). Each color refers to one transformation: square (red), cube (green), exponential (blue) and logarithmic (cyan). Note that the correlations are computed for 12 tumors on the test subset. The* ***black*** *line represents the median correlation of the model fitted with original variables.*

are fitted using the function **loess** in R from the package *stat*. For a given response variable y, one predictive variable x, a degree, and a span, the function fits a local polynomial at each point of the vector x using its neighbors to predict y. The protocol presented in algorithm 5 is repeated nine times, to cover the possible combination of the three polynomial degrees with the three considered percentages of nearest neighbors. In total, we have 18 models, 2 for each degree-span combination. Median Spearman correlation coefficients for each model are shown in Table 3.3.

First, all models fitted with loess transformed variables (no matter the parameters) show higher performances than the reference model fitted with original variables (median correlations = 0.5609). In addition, if we compare models with only transformed variables

Algorithm 5: Protocol of model fitting and evaluation for each type of transformations

Data: Gene expression in a tumor and nucleotide compositions in *training 1 subset* (y_{set1}, X_{set1}), and in *training 2 subset*(y_{set2}, X_{set2}).

Result: Two Lasso penalized linear regression fitted with i) transformed variables or ii) original and transformed variables.

Steps to apply for each tumor:

1. For each variable in X_{set1}, estimate the non-linear transformation (loess, hockey stick or segmented regression) of X_{set1} to explain y_{set1}
2. Apply the estimated non-linear transformation to X_{set2}. Let $F(X_{set2})$ be the new transformed variables.
3. Fit an ℓ_1 penalized regression model using only $F(X_{set2})$ the new transformed variables on *training 2 subset* (160 variables).
4. Fit an ℓ_1 penalized regression model using both original and transformed variables on *training 2 subset* (320 variables).
5. Evaluate both models on *test subset* (Spearman correlation).

(160 variables), a Lasso penalized regression model fitted with Loess transformation with degree 1 and 75%-nearest neighbors present the highest performances (median correlation = 0.5752) while the one fitted with degree 1 and 25%-nearest neighbors has the lowest performance (median correlation = 0.5611).

Furthermore, the model fitted with both original variables and loess transformations of degree 1 and an $\alpha = 75\%$ (median correlation = 0.5939) shows the highest correlations compared to all models evaluated in this section. The performances of this model are slightly higher than that fitted with original variables and square transformations (best model performances in Section 3.3.1 with median correlation = 0.583). Besides, we notice that when adding original variables to all the models fitted with loess transformed variables, the correlations of the different models increase of about 2% (see Table 3.3). This

	Degree 2		Degree 1		Degree 0	
Variables	Transformed	Original + transformed	Transformed	Original + transformed	Transformed	Original + transformed
Alpha = 75%	0.5751	0.5929	0.5752	0.5939	0.5749	0.5927
Alpha = 50%	0.5712	0.5935	0.5713	0.5933	0.5712	0.5935
Alpha = 25%	0.5613	0.5887	0.5611	0.5880	0.5612	0.5884

Table 3.3: ***Performances of the loess transformation*** *For each combination, the degree of the polynomial and the span, two median correlation are shown: one for the model fitted with only transformed variables and one for the model fitted with original and transformed variables. In red and blue, respectively the highest and the lowest correlation using only transformed variables. These correlations are compared to those of the reference model of median correlation equal to 0.5609.*

result indicates that even though loess transformed variables allow to retrieve an essential part of information for predicting gene expression, complementary information is still present in the original variables. One can conclude that we observe both a linear and a non-linear contribution of the nucleotide compositions for explaining gene expression.

3.3.3 Hockey stick regression

Figure 3.5 shows an example where the distribution of the median gene expression is represented in function of quantile of CpG composition in CORE promoter. To represent this distribution, we divide genes into ten groups of equal size in respect to the ten quantiles of their correspondent values of CpG_CORE, and for each group of genes, we calculated the median of gene expression in an Ovarian tumor. We can notice that there are two different trends before and after the 7^{th} quantile (bar at x = 0.071). This representation is similar for many nucleotide compositions. This is our motivation to use hockey stick regression and piecewise regression (see next subsection) to evaluate two different non-linear models

that predict gene expression in function of each one of the nucleotide composition. The distribution of the medians of gene expressions in functions of groups of nucleotide composition in Figure 3.5 can be seen as hockey stick model (linear model & constant model).

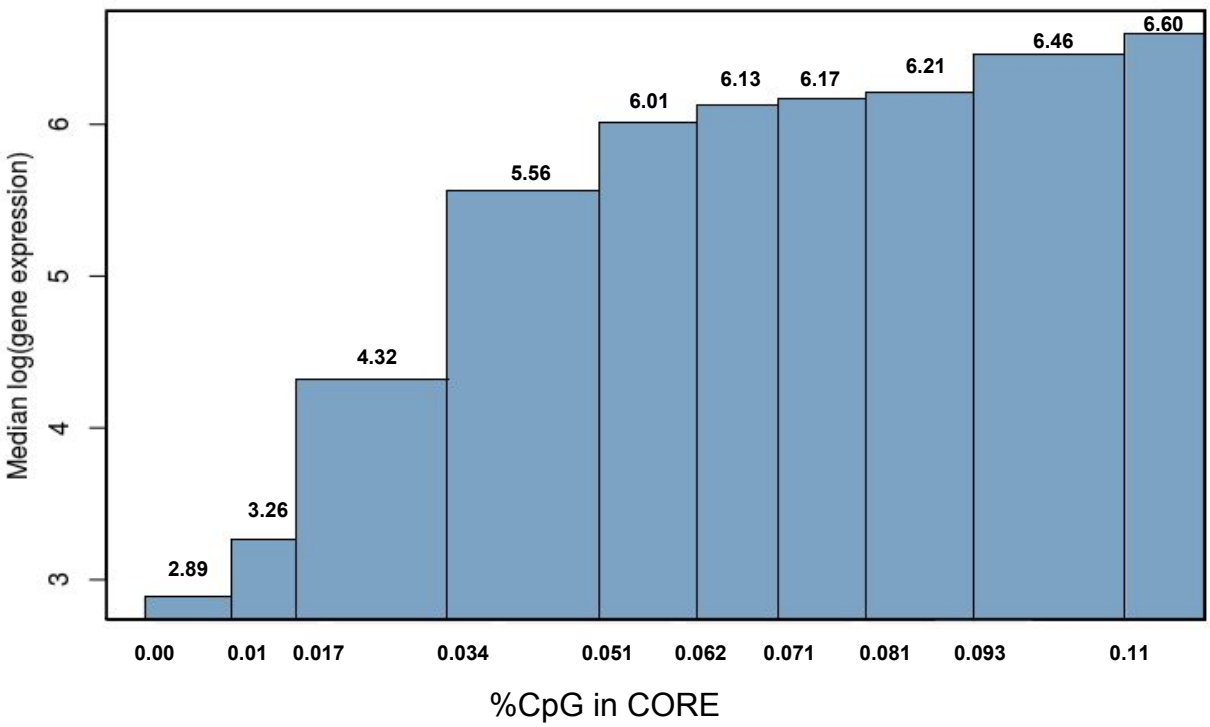


Figure 3.5: ***Distribution of the median of the log of gene expression in an Ovarian tumor in function of 10% quantiles of CpG in the CORE promoter.*** *Each bar represents the median of the log expression of genes belonging to each quantile groups of CpG in the CORE promoter. The values on the x-axis represents the different quantiles of CpG_CORE at respectively* $q = [10\%, 90\%]$. *The values of the median gene expression in each group are noted in the top of each bar.*

Hockey stick regression (see Section 1.2.3.2) is a non-linear model for predicting a response variable using two models, delimited by a breakpoint: i) a constant model, and ii) a linear model. As presented in Section 1.2.3.2, there are two types of hockey stick regression, Type I: the constant model for values of the predictive variable lower than the breakpoint, and Type II: the constant model for values of the predictive variable highest than the breakpoint (see Equation (1.19)).

As explained in our protocol for model fitting and evaluation (Algorithm 5), for each nucleotide composition, we learn a hockey stick regression transformation on *training 1 subset* then we fit two Lasso penalized regression models one with only hockey stick transformed variables and one using original and transformed variables on the *training 2 subset*. Note that when fitting hockey stick regression on *training 1 subset*, we select the best type (I or II) for each predictive variable. We proceed as follows. First, for each variable, we select a set of equidistant breakpoints between the quantiles 10% et 90%, and for each breakpoint, we learn two hockey stick models (type I and type II) on the *training 1 subset*. The mean square error is calculated for each model on the same subset. For each variable, the selected hockey stick transformation type is the one having the lowest mean square error. Then the selected hockey stick transformation is applied to the same variables in the *training 2 subset*. Finally, two comprehensive models are fitted: i) One using only hockey stick transformed variables and ii) one using original and transformed variables on the *training 2 subset*. Spearman correlation coefficients calculated on the *test subset*, for each tumor, are illustrated in Figure 3.6. However, fitting a model with both original and transformed variables shows higher performances (median correlation = 0.58) than the model fitted with only transformed variables (median correlation = 0.553). Potential reasons were presented previously (Section 3.3.2). Surprisingly, a model using only hockey stick transformed variables without original variables (median correlation = 0.553) does not outperform a model fitted with only original variables (median correlation = 0.560). Such a result may have several interpretations. One can suggest that, despite the median behavior that we observed in Figure 3.5, some important variables are not suitable for such form of transformation and thus lose their real ability to explain gene expression. Another explanation could be that the predicted mean square error computed on genes fitted with a constant model is very high. One can think that these genes are not constant and can be fitted with another linear model. From this observation, we consider fitting the genes with one another segmented transformation known as piecewise regression.

3.3.4 Piecewise regression

A piecewise linear regression (see Section 1.2.3.1) is the general form of linear segmented regression where, using the same concept as hockey stick, in order to predict a response variable y in function of a predictive variable x, two distinct linear models are fitted delimited by an estimated breakpoint (see Equation (1.14)).
The first step in our protocol (Algorithm 5) is to learn the transformation model. The steps of learning piecewise transformation on the *training 1 subset* are described in the algorithm 6. This procedure is applied to each variable of each tumor.

Algorithm 6: Piecewise transformation

Data: Gene expression in a tumor (y_{train}), nucleotide composition (X_{train}) in *training 1 subset*, and nucleotide composition (X_{model}) in *training 2 subset*

Result: Estimation of the piecewise transformation parameters (β and x_0) and a vector of transformed nucleotide compositions from *training 2 subset*

Function segmented of R:

1. Learn a linear model on *training 1 subset*: LM = **lm**(y_{train} X_{train})
2. Initialize the breakpoint: ψ= median(X_{train})
3. Fit the segmented model on *training 1 subset*: SEG = **segmented**(LM, X_{train}, psi)
4. Return the best segmented model (estimated coefficients and threshold)
5. Build transformed variable on *training 2 subset*: PRED = **predict**(SEG,X_{model})
6. Return the best model parameters: β and x_0 and the vector of new transformed variables: PRED

The algorithm is based on the function **segmented** from the R package **Segmented**. This function takes as an input a linear model fitted with the function **lm** to predict a response variable y in function of a predictive variable x. The breakpoint (noted in R as ψ) must be initialized. We use the default value that is the median of x. The function **segmented**

fit the model as explained in Section 1.2.3.1. The output of this function is the best model coefficients estimations (Equation (1.15)) as well as the estimated breakpoint (x_0). Finally, we apply the estimated a piecewise transformation to variables in *training 2 subset* and fit, on that same training subset, two Lasso penalized regression models with transformed variables and with transformed and original variables. Spearman correlation coefficients for each model calculated on the *test subset* on 2000 genes for 12 tumors are presented in Figure 3.6. A Lasso penalized regression model fitted with piecewise transformations only presents slightly higher performance (median correlation = 0.5694) than a Lasso penalized regression model fitted with only hockey stick transformation (median correlation = 0.5536), and then a Lasso penalized regression model fitted with original data (median correlation = 0.5609). When combining original and transformed variables, a model fitted with piecewise transformation present the highest performances with a median correlation of 0.5905.

3.3.5 Comparison with MARS

To evaluate our methods, we compare the performances of the best model, from each non-linear transformation to those of MARS: the Multivariate adaptive regressions splines (see Section 1.2.3.3). In this section, we fit a MARS model to explain gene expressions with all nucleotide compositions (160 variables) on *training 2 subset* using a forward selection without interactions (Section 1.2.3.3). The initial aspect that differentiates MARS from our approach is that MARS generates base functions by stepwise searching over all possible univariate candidate breakpoints and across interactions among all variables while in our approach we deal with each variable independently.

MARS model is fitted using the function **earth** from the R package **earth**. All the parameters are set to the default except the degree of interaction (equal to 1 by default *i.e* no interactions), we consider degrees of interaction equal to 1 and 2 where a degree equal to 2 refers to including second-order interactions.

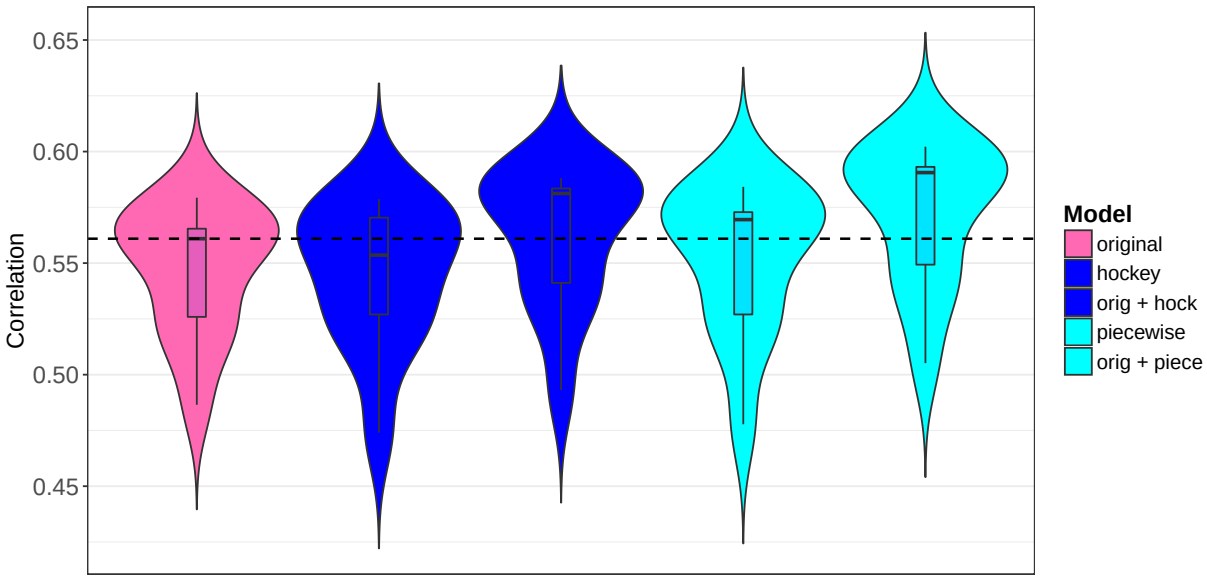


Figure 3.6: ***Performances of Hockey stick and piecewise transformations****. Colors respectively correspond to models fitted with: original variables (pink), hockey stick (blue) and piecewise transformation (cyan). For each transformation the first boxplot represents a model fitted only with transformed variables and the second to a model with original and transformed variables. The black horizontal line represents the median correlation of the model fitted with only original variables.*

Figure 3.7 illustrates Spearman correlations for the model fitted on original variables, and the models fitted with only transformed variables (160 variables) using the best estimation of the different methods: simple transformation (Square), loess regression (degree = 1, spam = 0.75), hockey stick regression, piecewise regression as well as those for MARS without interactions (degree=1). The performances of the model fitted with original variables (median correlation = 0.56) are higher than those of the MARS model (median correlation = 0.52). This can be explained by the strong regulation when fitting a MARS model (mean number of selected variables for MARS is equal to 19 whereas the reference model includes 160 variables). Besides, the MARS model is designed to include interactions and may present better performances then.

In this section, we present different non-linear variables transformations and evaluate the

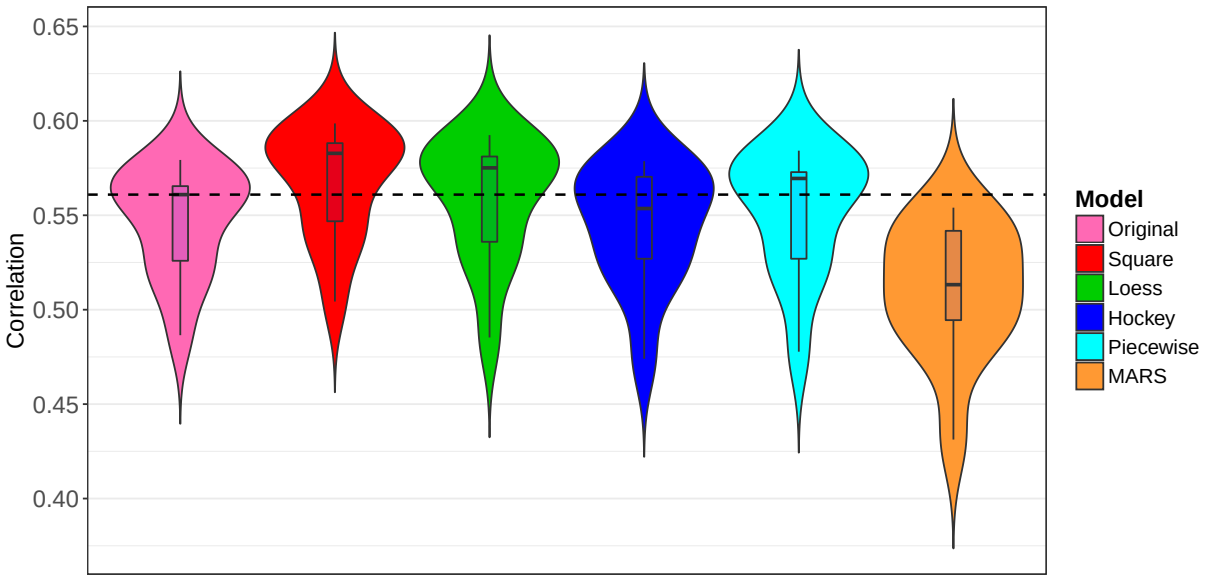


Figure 3.7: ***Performances comparison between best models of each non-linear transformation.*** *From left to right, boxplots represent correlations for a penalized model fitted on the training 2 subset using: original variables (pink), square transformed variables (red), loess transformation with a polynomial of degree 1 and 75% nearest neighbors (green), hockey stick transformation (blue), piecewise transformation (clear blue) and MARS with a forward selection where non interaction is included (orange). The black line is the median correlation of the model fitted with original data*

effect of non-linearity in predicting gene expression. Even though considering a non-linear relationship between gene expression and nucleotide compositions shows similar or slightly higher performances (maximum increase of 0.8% in correlations), this increase is limited. Also, adding variable interactions to the MARS model (with parameter degree equal to 2) should increase its performances. This can also be true for our models. For this reason, in the next section, we include pre-selected interactions of non-linear transformed variables using the chi-square test approach introduced in Section 3.2 and compare our results to the MARS model with interactions.

3.4 Including interactions of non-linear transformation

In Section 3.2, we prested that using a χ^2 test, allows us to detect second order interactions with a small increase in model performances. In addition, in Section 3.3, we presented different non-linear transformation models specially loess which seems to increase slightly model performances. In this section, we fit a model using transformed variables and pre-selected second order interactions calculated on transformed variables. This work can be compared to a MARS model (Section 3.3.5) including second-order interactions. For this aim, we proceed as described in Algorithm 7.

Algorithm 7: Non-linear transformations and interactions in linear models

Data: Gene expression in a tumor and nucleotide compositions in the *training 1 subset*, the *training 2 subset* and the *test subset*.

Result: A Lasso penalized linear regression model fitted with transformed variables and their pre-selected second order interactions.

Steps to apply for each tumor:

1. Transform the 160 variables with a loess regression for all the genes (16294 genes) with a first degree polynomial and 75% of nearest neighbors (see Section 3.3.2).
2. Run the interaction algorithm χ^2 test on the loess transformed data (see Section 3.2.1) using interactions defined by the minimum operator and a quantile of 2% as a threshold for the selection rule. The test is applied on the *training 1 subset*.
3. Run an ℓ_1 penalized regression model with transformed variables and pre-selected interactions for each tumor on the *training 2 subset*.
4. Apply and evaluate (Spearman correlation between predicted and real gene expression) on the *test subset*.

Figure 3.8 shows Spearman correlation for different lasso penalized linear regression fitted with: 1) original variables (Reference Model) 2) loess transformed variables (Section 3.3.2) 3) original variables along with their pre-selected second-order interactions (inferred in Section 3.2.3) 4) loess transformed variables along with their pre-selected second-order interactions (Section 3.4) and finally 5) MARS with a forward selection and with second-order interactions. Lasso penalized regression model fitted with loess transformations and their pre-selected second-order interactions presents the highest performances (median correlation = 0.5872). MARS model with second-order interactions presents slightly higher performances than a the model fitted with original variables (median correlation for MARS = 0.5666 Vs. 0.5606 for the original variables) but not higher than loess transformations.

In this Chapter, the initial aim was to add new variables (interactions) or change the model hypothesis (non-linearity) to increase model performances. Besides, we searched for different model combinations that can be significantly better than the reference model. Even though we obtained small improvements, they are not very significant, and they complicate the training process without considerable advantages.
However, predicting a response variable using non-linearity and including interactions between predictive variables, is not a new subject. Very advanced methods using deep learning exist with this aim. In the next Chapter, we study these models with different architectures to explore further the interest of non-linearity and variable interactions in predicting gene expression from the sequence.

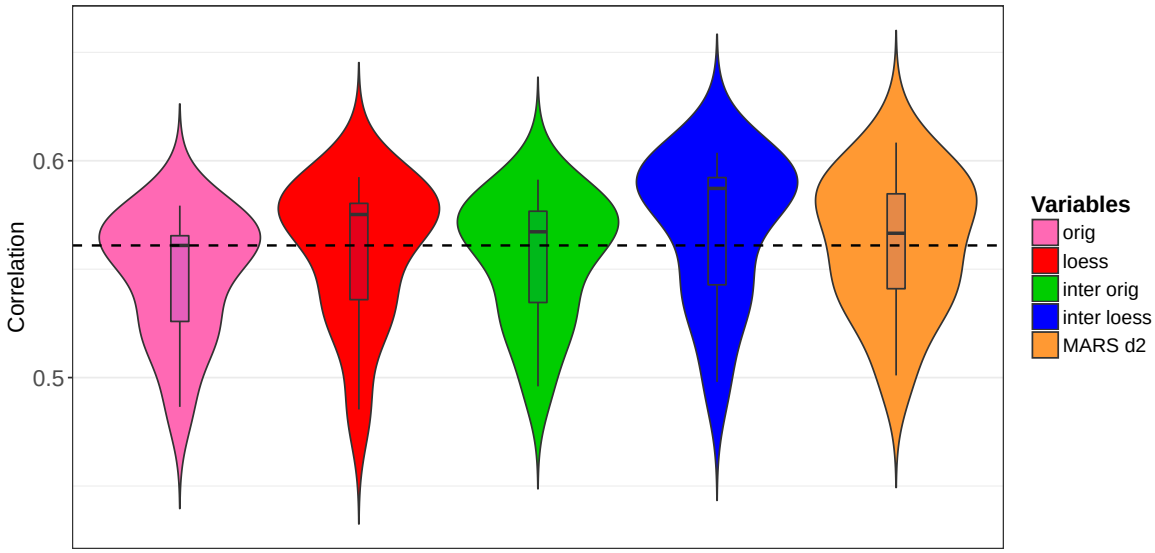


Figure 3.8: ***Performances of a model using interactions based on loess transformed variables.*** *The first four boxplot are the performances of an ℓ_1 penalized regression on different types of variables: original variables (pink), loess transformation with degree 1 and span of 75% (red), model with original variables along with the interactions between original variables selected with a χ^2 test with a min operator with the quantile 2% as threshold (green), model with loess transformed variables along with interactions between loess transformed variables selected with a χ^2 test with a min operator with the quantile 2% as threshold (blue). The final boxplot (orange) represents a MARS model fitted on original data with second order interactions. The black line is the median correlation of the models fitted with original data*

Chapter 4

Artificial neural networks

In this Chapter, we present the results for different artificial neural networks with the aim of predicting gene expression from DNA sequences. Two different problems are considered. The purpose, for both studies, is to predict gene expression but the types of predictive variables are different for each problem. The first study is a deep neural network based on the same predictive variables defined in previous chapters (nucleotide compositions) while the second problem is a convolution network based on the entire DNA sequences as predictive variables. This chapter is divided as follow, in Section 4.1 we present Keras library for artificial neural networks. A validation process required for both problems is shown in Section 4.2. In Sections 4.3 and 4.4, we present respectively, the optimization process and the results of both problems.

4.1 Keras

Keras is a high-level neural network API, written in Python and that uses Tensorflow, or Theano (not used in this thesis) as a backend. It was developed as part of the research efforts of the ONEIROS project (Open-ended Neuro-electronic Intelligent Robot Operating System) [C+15]. Its primary author and maintainer is François Chollet, a Google engineer. Keras contains numerous implementations of commonly used neural network building blocks such as layers, objectives, activation functions, optimizers, and many tools to make working with image and text data easier. The core data structure of Keras is a

model, *i.e* a way to organize layers. The simplest type of model is the Sequential model, a linear stack of layers. Four steps are required: stacking layers, compilation, training, and evaluation. Next, we present different functions with their parameters used in each of the steps.

1. **Step 1 - Stacking layers:** Create model architecture by adding different hidden layers. The different layers we use in this chapter are:

   - Dense: a fully connected layer. A necessity to specify the number of outputs and the activation function. Regularization and neuron constraint can be added.
   - Conv1D: convolution layer. A necessity to specify the shape of the input layer (if the first layer), the number of filters (kernels) and the length of these filters as well as the activation function. Regularization and neuron constraint can be added.
   - Maxpooling1D and Averagepooling1D: pooling layer using the maximum or the average functions, respectively. This layer comes after a convolution layer. Two parameters have to be tuned: the pool, *i.e.* size of the pooling window, and the stride, *i.e.* the factor by which to downscale. In general, the stride is equal to the pool.
   - Dropout: applies Dropout to the input. To initialize the fraction p of dropped neurons.

2. **Step 2 - Compilation step**: Used to configure the model for training. The *compile* function is used and requires to choose the optimization algorithm (RMSprop or Adam), the learning rate, and the cost function (mean square error for regression, see Section 1.3.3.2). After this step and before the training, additional parameters known as callbacks should be initialized if needed. One of the essential callbacks is the early stopping regularization with defined patience (see Section 1.3.5). An additional important callback is to monitor the evolution of model performances at each iteration and save the best model, the $ModelCheckpoint$ function is used and requires the name and the directory of the saved model.

3. **Step 3 - Training step:** The backpropagation algorithm. This step is applied using the fit function that requires the following entries:
   - Training data: the input (predictive variables) and output (predicted variable)
   - Validation split: percentage of examples of the training data used as the validation set.
   - Validation data: a subset of data (input, output) used for validation. This entry is used instead of Validation split.
   - Batch size: the mini-batch size for the gradient descent.
   - Number of epochs: number of forward/backward passes.
   - Callback (if required): Defined in step 2.
4. **Step 4 - Evaluation:** Apply the network to the test set using the best model obtained from the training in step 3. The *evaluate* function returns, for an input and output test set and batch-size, the mean square error. Additional evaluation of model performances is done by computing the Spearman correlation coefficients between the predicted and the observed output on the test set.

Note that training a neural network can take days or even weeks to converge, which can be hugely expensive. To overcome this problem, we used a Graphics Processing Unit (GPU) valid for the $tenserflow-gpu$ version used by keras (https://github.com/mind/wheels).

4.2 Validation procedure

As in the previous chapter, an independent set of genes is used for assessing the different models. From the initial dataset, we randomly select a set of 2000 genes (*i.e.* individuals) noted hereafter as the "test set". This subset is only used for evaluation and model comparison and will not be included for training and validation of the neural network. The remained individuals of the dataset are then divided for training and validation either by a percentage of data (80%- 20%) or by a random selection of a defined number of individuals for the validation set (see Section 1.3.3.3).

In this chapter, as well as in chapter 3, all models are fitted for 12 tumors chosen randomly from 12 different cancer types.

4.3 Predicting gene expression from nucleotide compositions

In this Section, the objective is to fit **multilayer perceptrons** and/or deep feed-forward networks to predict gene expression (output layer) in function of nucleotide compositions (input layer). The problem is equivalent to that studied in previous chapters, using the same response and predictive variables. The model trained in this section is a regression network with continuous variables. As presented in Section 1.3.3, this model is trained using the gradient descent optimization. We first present a procedure to optimize model parameters (Section 4.3.1). Then the selected model with the highest performances (in term of Spearman correlation) is compared to a Lasso regression model fitted on the same data frame (Section 4.3.2).

The model is fitted for each tumor to predict gene expression (response variable) in function of nucleotide compositions in different regulatory regions (160 predictive variables) using 16,298 genes (number of individuals). Before training, predictive variables are normalized as for the Lasso penalized linear regression fitted with the glmnet function from R (see Section 1.2.1.2).

4.3.1 Optimizing network architecture

A neural network has a certain number of parameters that must be initialized before training, such as the number of layers and of neurons, the optimizer, the regularization, and others (see Section 1.3.6). The choice of each parameter depends on the dataset and should be chosen to induce the maximum improvement of the model performances.
In this section, starting from a set of parameters, we proceed in a forward study where, at each step, only one parameter is tuned (other parameters are conserved from the previous step), and its best value is retained for the next steps. Let P be the parameter to tune at

one step, and let $\{p_j\}_{j\in 1,\dots,M}$ be the set of M different values of P. M different networks, one with each p_j, are trained and then evaluated on the test set. The value of P that corresponds to the highest Spearman correlation between observed and predicted gene expression is retained. We detail below the different steps used in this study for optimization (see Figure 4.1).

1. **Initialization step:** We start the training by a model with the nucleotide compositions of a gene (160 predictive variables) as input layer and the log of the gene expression as fully-connected output layer (one neuron) with a linear activation function. In addition, we initialized the following training parameters:

   - Optimizer: Adam with learning rate $\eta = 0.001$
   - Number of iteration: epochs = 1000
   - Data partition: validation split = 20%
   - Mini-batch: batch size = 200
   - Early stopping: patience = 100
   - Each layer is followed by a dropout layer with a fraction $p = 0.4$

2. **Step 1:** Tuning the number of layers and the number of neurons per layer (Section 4.3.1.1). The number of layers/neurons that present the higher performance are selected and are called *architecture 1* in the following.

3. **Step 2:** Tuning the optimizer (Section 4.3.1.2). A new network is fitted while retaining all the parameters of *architecture 1* except the optimizer. The *architecture 2* represents the network with the best optimizer.

4. **Step 3:** Tuning training parameters (Sections 4.3.1.3 and 4.3.1.4). In this step, one by one, we select the type of regularization (dropout, ℓ_1 or ℓ_2) as well as the batch size and the early stopping patience. At the end of this step, we obtain the optimized architecture, *i.e* the one that provides the highest Spearman correlation between observed and predicted gene expression on the test set.

Note that the number of epochs, as well as the validation split, are conserved over all the optimization process. Finally, each network is fitted for 12 tumors and evaluated (correlations) on the test set of 2000 genes.

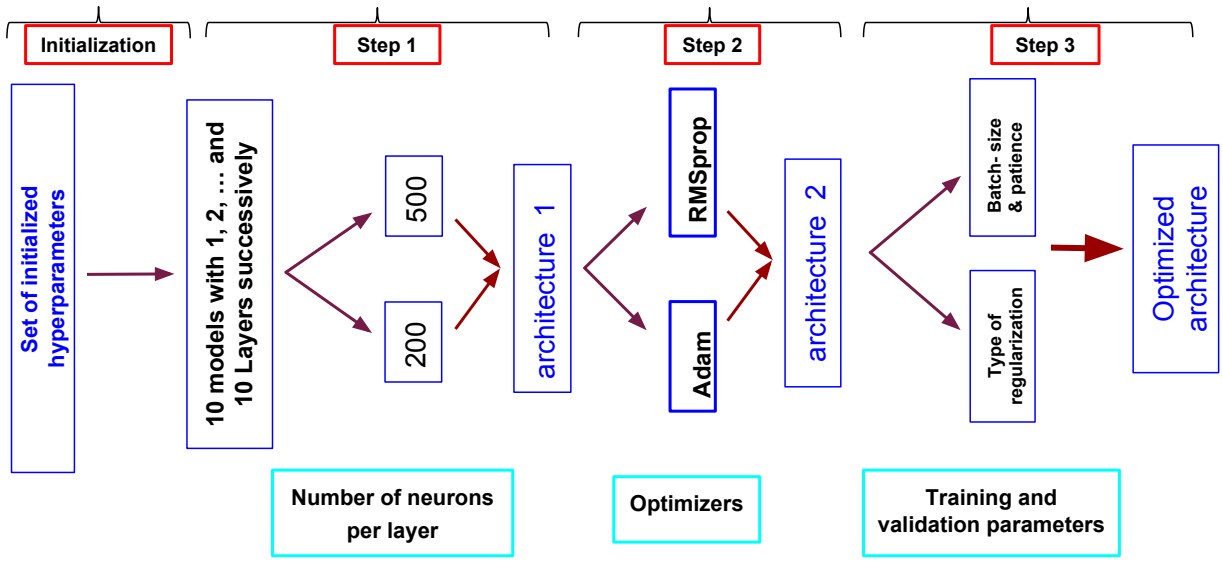


Figure 4.1: ***Steps of architecture optimization.*** *hyperparameters to tune. From left to right: number of layers, number of neurons, optimizers, training and validation parameters.*

In the following, we present the different values tested for each parameter, the Spearman correlations of the corresponding networks, as well as the best architecture obtained.

4.3.1.1 Number of layers and neurons

One critical parameter to be tuned in all neural networks models is the number of hidden layers and the number of neurons in each layer. In our networks, all layers are fully-connected (Dense function). The number of layers and neurons in a network highly depend on the dimension of the trained data and on its complexity. In addition to the number of neurons, each hidden layer is characterized by an activation function (see Section 1.3.2.2). In this study, all hidden layers have a ReLU activation function. With the aim of selecting the best number of layers and neurons for this regression problem, we define a set of possible values for the number of hidden layers $\{L\} = [1, 10]$ and a set of neuron numbers

per layer $\{N\} = \{200, 500\}$. Note that we only consider the case where all layers in the same network have the same number of neurons. Furthermore, for each layer, the weight matrices are initialized with a random uniform distribution. For each l in L and n in N, we fit a network with l layers and n neurons per layer. All other parameters are used as initialized. In total, we fit 20 networks, 2 for each number of layer. Table 4.1 displays the median Spearman correlations of all fitted networks. Columns represent the number of hidden layers in the network, and rows indicate the number of neurons in each layer. First, when comparing rows, we observe that the median correlations vary very slightly between networks with 200 neurons or 500 neurons per layer. Besides, in some models (with four layers for example) the performance with 500 neurons per layer is lower than that with 200 neurons per layer (median correlation = 0.617 vs. 0.624).

On the other hand, if we compare the performances of the model when adding more layers, we notice that the model with only one hidden layer shows the lowest performances. As we go deeper in the model (*i.e* adding more hidden layers), the performances of the networks increase in term of correlation until they reach a maximum of 0.628 for a network with five layers and 200 neurons in each. Adding more than five hidden layers does not improve predictions, on the contrary, we notice a slight decrease in Spearman correlation. This decrease may be explained by the fact that deeper networks require the training of a larger set of parameters, hence require a larger set of data.

	1L	2L	3L	4L	5L	6L	7L	8L	9L	10L
200N	0.597	0.618	0.616	0.624	0.628	0.620	0.620	0.614	0.615	0.611
500N	0.596	0.618	0.617	0.619	0.627	0.613	0.623	0.616	0.613	0.610

Table 4.1: ***Comparison between the number of layers and the number of neurons.****. Each column represents the number of ReLU hidden layers in the fitted network, and each row represents the numbers of neurons in each layer. Correlations are computed on the test set (2000 genes)Median Spearman correlation coefficients over 12 tumors.*

As explained above, the number of hidden layers and the number of neurons that induce the highest correlation are selected for the following optimization. We define *architecture 1*

as the network built with five hidden layers with ReLU activation function and 200 neurons each. All other parameters are retained from the *Initialization step*.

4.3.1.2 Optimizers

In Section 1.3.3.2, we presented two different optimizers, Adam and RMSprop, that both require initialization of the learning rate that is modified during the training in function of the gradient. In this section, we will compare the performances of both optimizers. Note that the initialization of the learning rate is set to $\eta = 0.001$ for both optimizers. Figure 4.2 illustrates the Spearman correlations of the network fitted with Adam (cyan), and fitted with RMSprop (purple). The median correlation of the network fitted with RMSprop over 12 tumors is 0.617 while that of Adam is 0.628. The Adam optimizer seems to outperform RMSprop in this study. At the end of step 2, the *architecture 2* is the same as *architecture 1* with five hidden ReLU layers with 200 neurons each and an Adam optimizer.

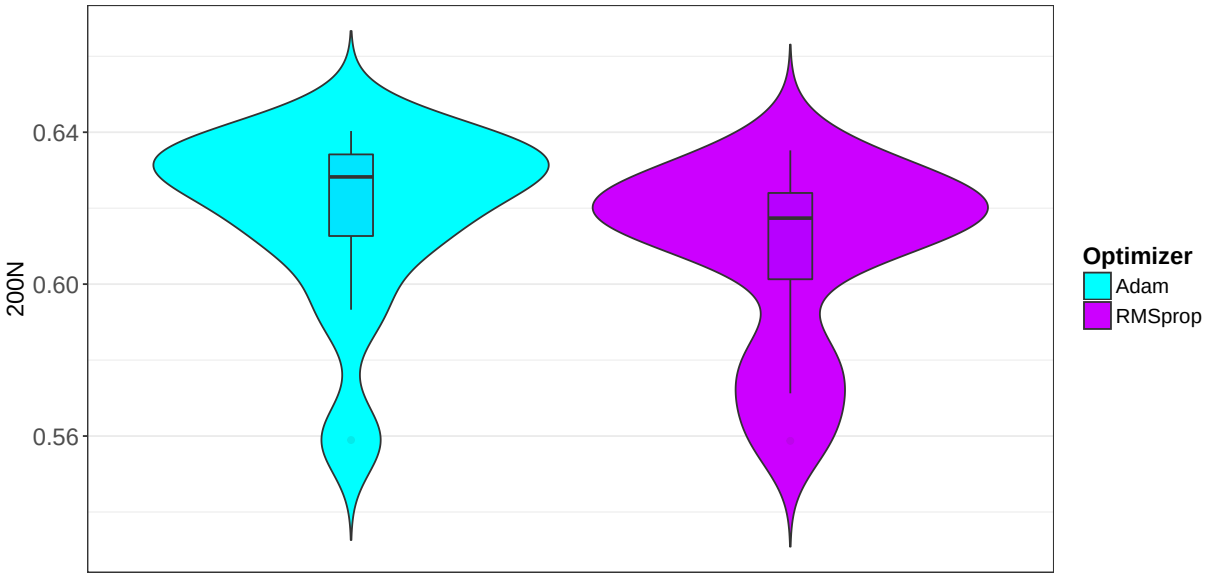


Figure 4.2: ***Tuning the optimizer.*** *The first boxplot (cyan) represents Spearman correlations of the network with* architecture 1*, the optimizer used in this architecture is Adam. The second boxplot (purple) represents the Spearman correlation of a network with the same parameters of* architecture 1 *except for the optimizer that is RMSprop.*

Note that the number of layers and of the number of neurons in each layer were tuned using the Adam optimizer (see Section 4.3.1.1). Hence it is possible that the performances presented above are biased. To ensure that this is not the case, we repeat the procedure of the previous section (*i.e.* tuning the number of layers) but only with 200 neurons by layer using the RMSprop optimizer. For each number of layer in $\{L\} = [1, 10]$, we fit a network with L layers of 200 neurons each. Figure 4.3 illustrates a comparison between the optimizers. For each number of layer L, two boxplots are showed corresponding respectively to the network fitted using Adam (cyan) and the network fitted using RMSprop (purple). We notice that using RMSprop as optimizer provide almost systematically slightly lower correlations than with Adam. The model with five hidden layers and 200 neurons in each has the higher performances with both optimizers. This result shows that the Adam optimize seems better for this problem. In the following, we proceed using the model fitted with Adam optimizers noted as *architecture 2*.

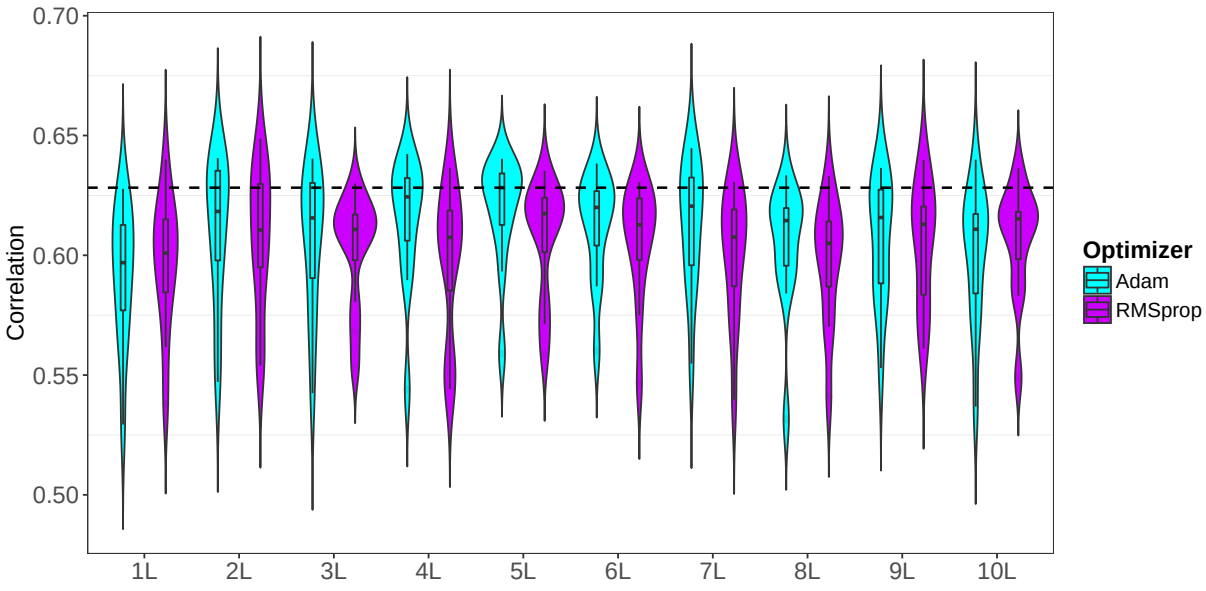


Figure 4.3: ***Comparison betwen Adam and RMSprop optimizers.*** *Spearman correlation for each network computed on the test set. For each number of layer* $\{L\} = [1, 10]$ *with 200 neurons each, two networks are fitted: i) using Adam optimizor (cyan) and ii) using RMSprop optimizer (purple). The dashed horizontal line represents the median correlation of the network with* architecture 2 *with 5 layers of 200 neurons each and Adam optimizer.*

4.3.1.3 Weight regularization

In the third and final step, we optimize the parameters used to avoid overfitting in the training process. First, we start by selecting the type of weight regularization, *i.e* choice between dropout and ℓ_1/ℓ_2 regularization. Secondly, we optimized the gradient descent regularization, *i.e*. the mini batch-size and the patience of the early stopping.

In Section 1.3.3.3, we presented two methods of regularization applied at each layer: i) dropout layers, and ii) ℓ_1/ℓ_2 regularization. Each regularization has a parameter to be optimized: i) fraction p for dropout, and ii) regularization parameter λ for ℓ_1 and ℓ_2. In this section, we aim to select the best form of regularization as well as the best parameter value. The previous model denoted *architecture 2* was fitted using dropout layers with fraction $p = 0.4$ following each layer. First, we proceed by optimizing the fraction p of dropout. Let $S = \{0, 0.5, 0.6\}$ the set of values of p. We fit three different networks with the same exact parameters as *architecture 2*, except for the dropout fraction where for each model we change $p \in S$. Note that $p = 0$ refers to a network with no regularization. The Spearman correlations of the three models are presented in Figure 4.4 (boxplots in purple). We notice that a model with no regularization ($p = 0$) presents low performances compared to other models and hence that regularization is useful here to avoid overfitting. For $p = 0.5$ and $p = 0.6$, the correlations are slightly lower than those of $p = 0.4$. Hence, for dropout, the best fraction seems to be 0.4.

In a second time, we study the effect of ℓ_1 and ℓ_2 regularization. Let $\Lambda = \{0.001, 0.01\}$ be the set of values that λ can take. Using each regularization, we fit two models, one for each value of λ. These models have similar parameters as *architecture 2* except that we use a regularization parameter instead of dropout. Spearman correlations for each model are summarized in Figure 4.4: i) green boxplots for ℓ_1 and ii) orange boxplots for ℓ_2. The ℓ_2 regularization provide low performances, with median correlations of 0.571 and 0.580 for $\lambda = 0.001$ and $\lambda = 0.01$ respectively. Performances of the networks regularized by ℓ_1 are higher than those with ℓ_2 but do not defeat the dropout regularization with a fraction $p = 0.4$: median correlations equal to 0.625 and 0.61 for $\lambda = 0.001$ and $\lambda = 0.01$

respectively, versus a median correlation of 0.628 with dropout. However, this result may be due to the value of the parameter λ since not all possible values were tested due to lack of time. Furthermore, one can think of using both dropout and ℓ_2 regularization. However, Krizhevsky & al. [KSH12] showed that dropout tends to have a much stronger effect than ℓ_2 regularization which hides the impact of the ℓ_2 regularization. Furthermore, using a dropout with an ℓ_1 regularization may lead to errors in the training process when at one layer, all weights are set to zero at a given iteration. An interesting perspective for the future would be to try a model with double regularization. We should also consider changing the parameter λ for ℓ_2 regularization to test if that increases the performances.

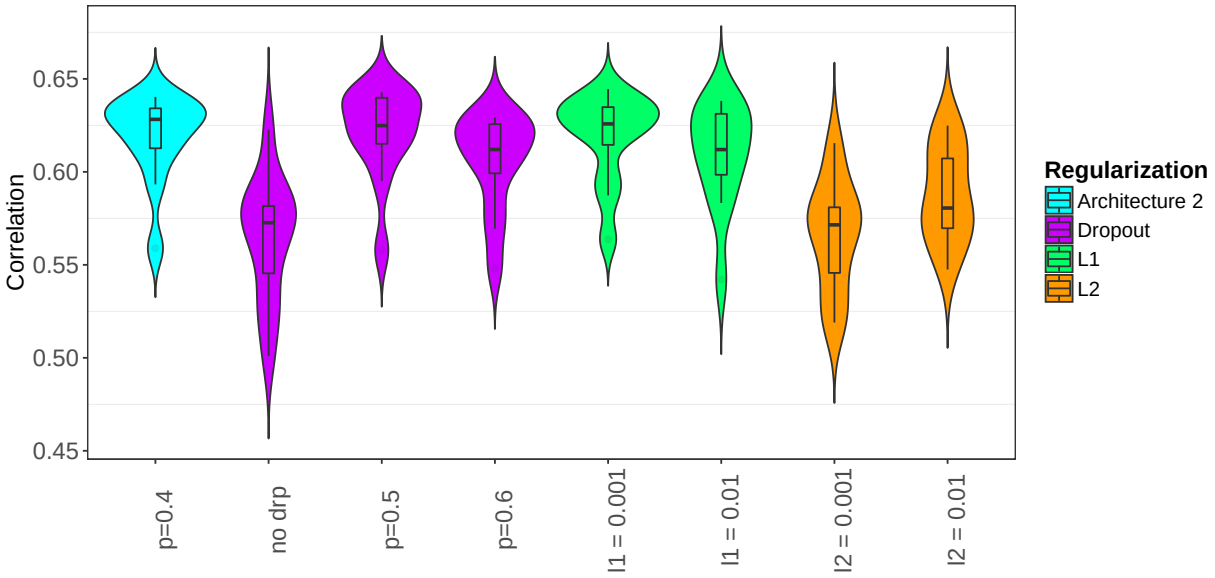


Figure 4.4: ***Comparison between regularization approaches and their parameters.*** *The first boxplot (cyan) refers to* architecture 2 *with dropout fraction p =0.4. For the other boxplots, each color represents a type of regularization and for each type each boxplot represents a model fitted with a different parameter's values: 1) dropout (purple) with* $p = \{0, 0.5, 0.6\}$*, ii)* ℓ_1 *(green, refereed to as L1), and iii)* ℓ_2 *(orange, refereed to as L2) with* $\lambda = \{0.001, 0.01\}$ *for each (l1 and l2 represent* λ*).*

4.3.1.4 Batch-size and patience

At this stage of the optimization, the *architecture 2* presents the higher performances. It is built with 5 ReLU hidden layers with 200 neurons in each and all the parameters from *Initialization step*, more precisely with a batch size of 200 and patience for early stopping of 100 iterations. In this last part of optimization, we tune these two parameters.

First, the batch size is the number of examples used to calculate the gradient at each iteration. The size of the batch depends on the number of individuals in the study. To cover different possibilities, we fit a model with a batch-size equal to 100 and another model with a batch-size equal to 300 (lower and higher than 200). All other parameters of the networks are conserved. Spearman correlations of the three different models with different batch-size are showed in Figure 4.5. A batch-size equal to 100 presented similar performances as a batch-size equal to 200, and a very slight decrease is noticed with a batch-size equal to 300 (median correlation = 0.623). In conclusion, the batch-size in this study does not have a strong influence on model performances, and hence we continue with a batch-size equal to 200.

Finally, with *architecture 2*, the final parameter to tune is the patience (see Section 1.3.3.3). A small value of the patience may lead to underfitting and not reaching the stopping point (Section 1.3.3.3), while a high value leads to overfitting. *architecture 2* was fitted with a patience $pa = 100$. Let $\{E\} = \{30, 50, 150\}$ be a set of values that can be attributed to the patience pa. For each $pa \in \{E\}$, we fit a model while maintaining all other parameters from *architecture 2*. Figure 4.6 shows the Spearman correlation coefficients for each fitted model. The first boxplot illustrates the network with *architecture 2*. A first conclusion is that the correlation with a $pa = 150$ is slightly lower than other values of pa with a median correlation of 0.624, indicating a possibility of overfitting. Correlations computed for networks with $pa = 30$ and $pa = 50$ are higher than that computed with $pa = 100$ with respective median correlations of 0.633 and 0.629. In conclusion, the patience of the early stopping has not a huge effect on the network. However, choosing small patience will reduce the time of execution for each model. The optimized model is that with patience equal to 30.

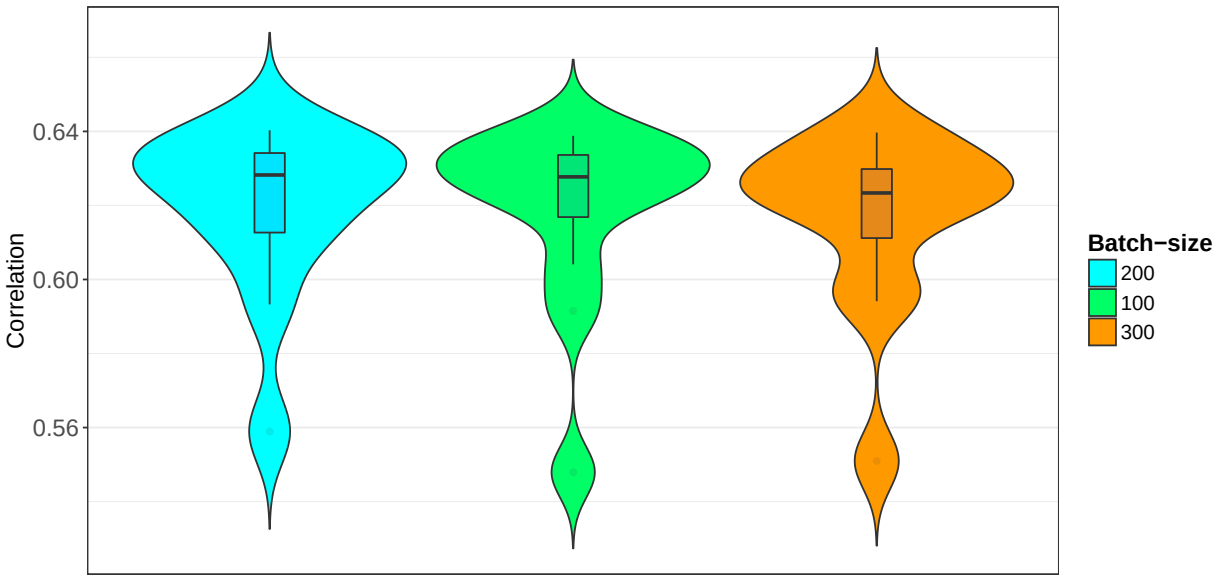


Figure 4.5: ***Batch-size optimization.*** *The first boxplot refers to correlations of* architecture 2 *with batch-size = 200. The two other boxplots represent correlations for networks with batch-size equal to 100 and 300 respectively. Each boxplot contains 12 values of correlation (one for each tumor) computed on the test set.*

4.3.1.5 Summary

Table 4.2 summarizes the optimization process. For each parameter (rows) the first column shows the set of tested value tested and the second column give the parameter retained for the optimized architecture. The final architecture involves:

- 5 hidden layers with ReLU activation function and 200 neurons with random initialization.
- Adam optimizer with a learning rate initialization $\eta = 0.001$
- Dropout layers following each layer with a fraction $p = 0.4$
- A batch-size of 200 example and an early-stopping with a patience of 30 iterations.

Our procedure to select the best architecture was based on the Spearman correlation between observed and predicted gene expression, calculated on an independent test set.

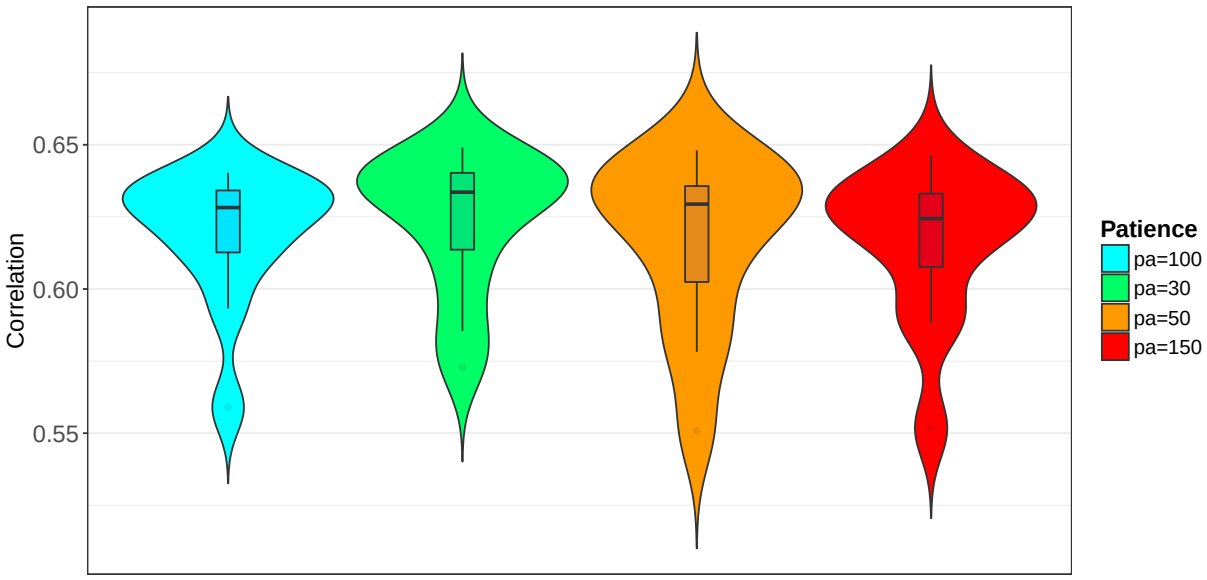


Figure 4.6: ***Tuning the early-stopping parameter.*** *Spearman correlation computed on the test set for 12 tumors. Each boxplot represent a network fitted with parameters of* architecture 2 *except for the patience that is for each boxplot from left to right: 100, 30, 50, and 150.*

One can also evaluate the model during the training algorithm by tracking the evolution of the mean square error (see Section 1.3.3.3). Figure 4.7 illustrates for one tumor (an Ovarian patient) the path of the error during the process on the training set (in blue) and on the validation set (in green) at each epoch. The error on the training set is always decreasing while the error on the validation set is fluctuating. At a certain epoch, the error on the validation set reaches its minimum value. This is the epoch where the model weights are optimized and retained for predicting new data. This epoch is not necessarily the last epoch the training process pursues until reaching the stopping point after *pa* (30 in this example) iterations with no improvements.

Finally, this procedure does not lead to the best-fitted network (all the parameters are optimized). Some of the presented results may be biased by the fact that each hyperparameter is tuned independently without taking into considerations the inerrant dependencies between parameters. This bias can be reduced using different strategies (more details are presented in Section 4.5).

	Set of tested values	Optimized architecture
Number of layers	1, 2, 3, 4, 5, 6, 7, 8, 9, and 10 layers with ReLU activation	5 layers
Number of neurons per layer	200 neurons or 500 neurons	200 neurons per layer
Optimizer	Adam or RMSprop	Adam
Regularization	i) Dropout with p={0,0.4,0.5,0.6} ii) L1 and L2 regularization with λ = {0.001,0.01}	Dropout with p=0.4 following each layer
Batch-size	100, 200, and 300 examples	200 examples
Patience (stopping)	30, 50, 100 and 150 iterations	30 iterations

Table 4.2: ***Summary of the optimization process.*** *For each hyper-parameter, the first column presents all the correspondent values tested while in the second column, the retained values of each parameter, i.e. those used in the optimized architecture, are shown.*

4.3.2 Comparison with Lasso penalized linear regression

In chapter 2, we used a linear regression model with Lasso penalization to predict gene expression from nucleotide compositions, and we observe that the model often exhibits good performances. In this chapter, we trained a multilayer neural network for the same purpose. In this section, we compare the performances of these two approaches in term of prediction as well as in term of model complexity.

Figure 4.8 shows the Spearman correlations for the Lasso penalized linear regression (blue) and the multilayer network (green) with the architecture retained in the previous section. The neural network presents a higher performance in term of prediction than the Lasso penalized linear regression with a median correlation of 0.633 versus a median correlation of 0.614 for Lasso.
On the other hand, the Lasso penalized linear regression is a simpler model both in term of complexity (number of parameters) and training time. The mean time of execution of a

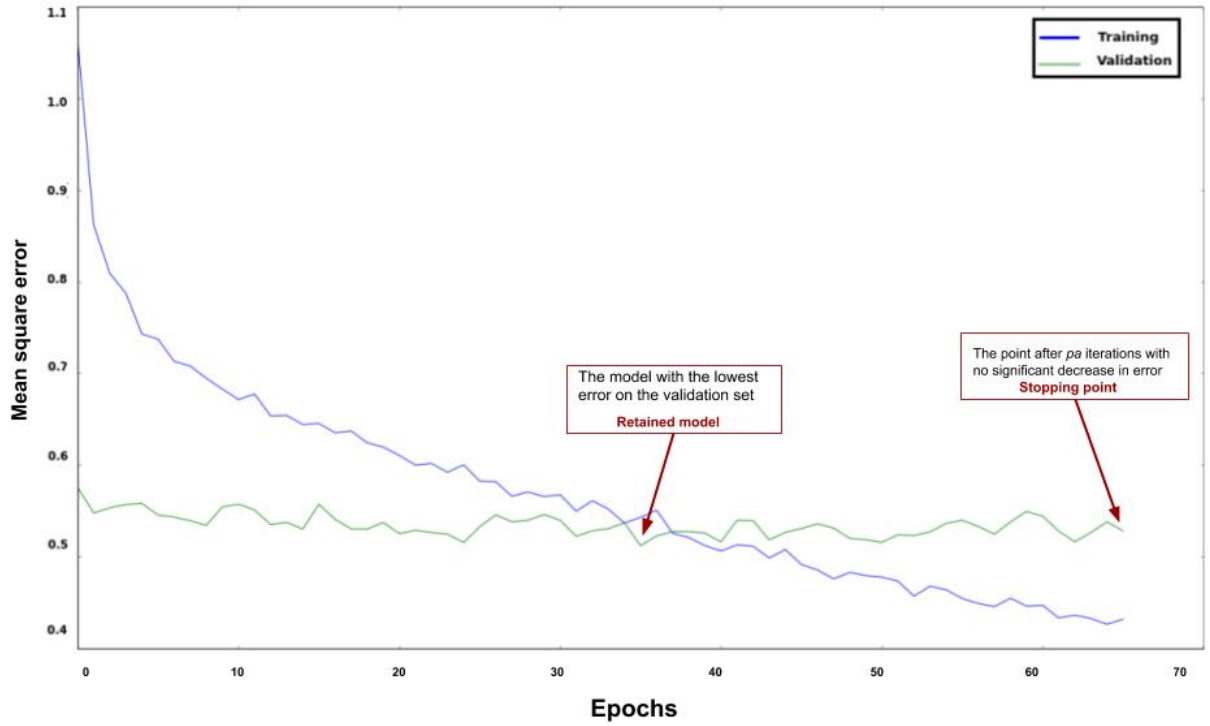


Figure 4.7: ***Graph of the evolution of the error*** *on the training and the validation set for the network optimized above. The graph refers to an ovarian tumor. The retained model and the stopping point represent the minimum of the validation error and the epoch of early stopping respectively*

Lasso penalized linear regression for one tumor is around 7 seconds on a standard computer while for the neural network, it can vary between 45 to 60 seconds depending on the architecture and on the GPU. More importantly, the Lasso penalized linear regression has an advantage in the term of variable explanation (important variables are easily identified) while the neural network is more like a black box. However, there is some interesting work in development for identifying the most important variables of the neural network (see Chapter 5 for further details).

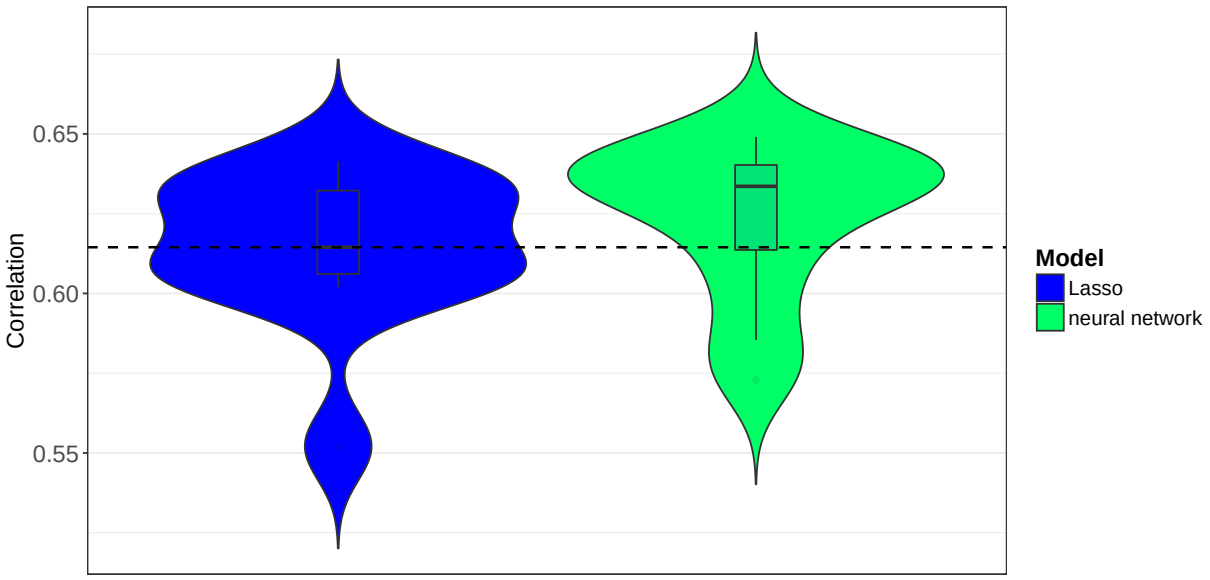


Figure 4.8: ***Comparison between multilayer network and Lasso penalized linear regression model.*** *Both models are fitted on the same training set with 160 variables and* $14,298$ *individuals and then evaluated (correlations) on a test set of 2000 genes. The first boxplot is the correlations of the Lasso penalized linear regression (blue) and the second refers to correlations of a multilayer neural network optimized above (green). The horizontal dotted line is the median correlation of the Lasso penalized linear regression.*

4.4 Convolution neural network

In all what we presented before, our models are based on a summary of DNA sequences defined by the nucleotide compositions and/or motifs scores (model from Chapter 2) in different regulatory regions as predictive variables. Even though these variables allowed us to achieve quite a good prediction accuracy, one can assume that resuming kilobases of DNA sequence with dozens of variables induces a certain loss of information. Hopefully, a one-dimensional convolution network can be fitted directly on DNA sequences. The architecture and the hyperparameters of the convolution network are presented in Sections 1.3.5 and 1.3.6. In this section, we present the pre-processing of the data (DNA sequences and PWM) and the iterative procedure we use to select the best architecture and the best choice for each parameter. Finally, we compare our model with the model fitted with linear

regression.

4.4.1 Global architecture and data pre-possessing

For this model, we use two types of data: i) DNA sequences of the CORE promoter for each gene, and ii) the position probability matrix for each motif given by the Jaspar database.

4.4.1.1 Input and output layers

The convolution network is used to predict gene expression from the DNA sequence of the promoter. Following the input layer, come a convolution and a pooling layer. In this study, all networks have only one convolution/pooling layer that aims to detect the presence of certain motifs in the DNA sequence (see Section 4.4.3 for more details). The output layer is a fully connected layer with linear activation function and one neuron representing the gene expression. These layers are retained all over the study, and additional layers can be added next.

Input layer

In this chapter, we use only CORE promoter sequences for 19393 genes. Motifs are mostly located on this sequences spread on -500/+500 base around the TSS. Each gene is characterized by a sequence of 1001 bases of A, C, G, and T. The adequate form of input for the 1D-convolution is a hot coding matrix, *i.e.* a 4-raw binary matrix with the number of columns equal to the length of the sequence (1001 bases). Each column is a binary vector of length four denoting the existence of each of the four nucleotides. Figure 4.9 shows the transformations of the CORE promoter sequence of each gene into hot coding matrix.

Output layer

This method aims to predict gene expression. The output layer is the expression of a gene in a tumor. Gene expression is downloaded from the TCGA Data Portal (see chapter 2). Each network was fitted for 12 tumors from 12 different types of cancer (see Section 3.1).

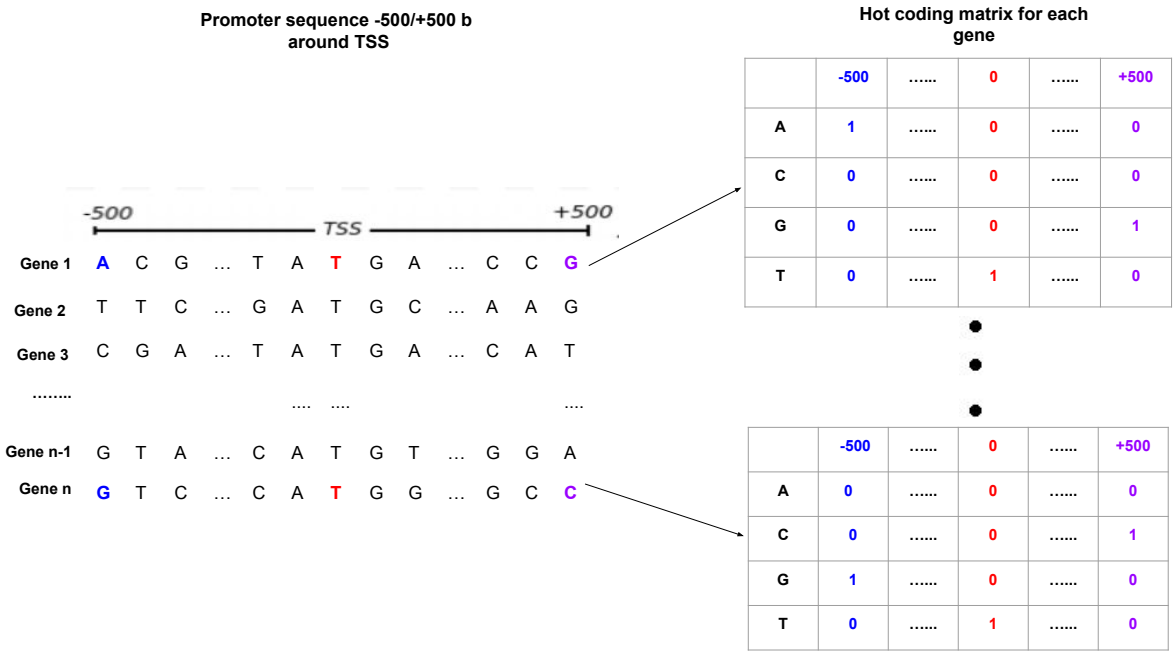


Figure 4.9: ***CORE promoter sequence transformed into hot coding matrix.*** *Each gene sequence of 1001 bases corresponds to a hot matrix of dimension* 4×1001. *For Gene 1, the base at position -500 is A (in blue) hence the first column at the hot coding matrix is* $\{1, 0, 0, 0\}$ *(first matrix in blue).*

4.4.1.2 Convolution layer

For the convolution layer, we consider initializing the filters with known filters: the motifs. In Chapter 1, we presented a probabilistic model to define motifs (see Section 1.1.3.1). In this thesis, we use both position probability matrices (PPM) and position weight matrices (PWM) to initialize weights. The matrices were collected from the Jaspar 2018 database [KFS+17]. The database provides PPM for 638 known motifs in the human genome. The length of motifs varies between 5 and 30 bases with a median of 11 bases. These motifs represent the filters in the first convolution layer.

Most motifs (550) have a length lower or equal to 15 bases. The 88 other motifs have a length between 16 and 30 with a median of 18 bases and are often a combination of two smaller motifs. In this study, motifs are restrained to those with a length lower or equal to 15.

Keras impose that filters in a convolution layer all have the same length (see Section 1.3.5.2). Hence for each motif with a length w lower than 15, we add $n = 15 - w$ columns with a uniform distribution (the sum of each column is equal to 1).

For each motif, we also calculate the PWM from the PPM using the following equation:

$$W_{b,i} = log\left(\frac{P_{b,i}}{p(b)}\right) \tag{4.1}$$

where $W_{b,i}$ is the value of PWM for base b and position i, $P_{b,i}$ is the value of PPM for base b and position i and $p(b)$ is the prioir probability of base b, *i.e.* the proportion of b in the CORE promoters. Note that we correct the position probability matrix by a pseudocount number equal to 10^{-5} in order to avoid zeros. According to what is done for PPM, we complete each PWM by null (0) columns in order to have filters of the same length.

4.4.2 Hyper-parameter optimization

The experiments are organized as follows. 2000 genes are retained as the test set for the model evaluation (correlations). Among the remaining 17393 genes, we randomly select 4000 genes used for validation (see Section 1.3.3.3 for details). All networks are hence trained on 13393 genes.

4.4.2.1 The first model

With the aim to optimize hyperparameters, we will start with a model called *model 1* that will be used as a reference in the following sections. As a start, we trained *model 1* with a set of hyperparameters (see below) and then, one by one, we test the effect of changing one hyper-parameter. When a tested value induces better performance, it is retained for the next steps. Note that some parameters are not examined in this thesis and will be maintained all over the study. The following layers and parameters constitute the *model 1* architecture.

- **Convolution layer:** 550 filters (number of PPMs) of size 15 bases each, and a ReLU activation function. Filters are initialized with PPM matrices (see Section 4.4.1.2).

- **Pooling layer:** global maximum over all the feature map (Section 1.3.5.3).
- Dropout layer with $p = 0.4$.
- **Dense layer:** fully-connected with 2000 neurons and ReLU activation.
- Dropout layer with $p = 0.4$.

Note that all the networks are trained using RMSprop optimizer with a learning rate $\eta = 0.001$, a batch-size equal to 200, 1000 epochs and with an early stopping of patience = 30. These hyper-parameters were not optimized for time reason.

To evaluate the model we apply the network on the test set and calculate the Spearman correlations for each tumor. The median correlation of *model 1* is 0.486. The boxplot of the correlations is presented in Figure 4.10 (first boxplot).

4.4.2.2 Effect of the weight initialization

First, we compared the performances of the network when changing the initialization of the weights on the convolution layer. For that we fitted two other networks, having the same architecture as *model 1* except that weights where i) initialized by PWM matrices (see Section 4.4.1.2) or ii) initialized with random uniform values $\in [0, 1]$.
The correlations of each network calculated on the test set are presented in Figure 4.10. The first boxplot (cyan) represents the accuracy of *model 1*. We notice that using PWM matrices or uniform random values as weights initialization decreases model performances (median correlation = 0.454, and 0.476 respectively). Hence *model 1* is retained after this step.

4.4.2.3 Effect of the pooling method and window size

The pooling layer in *model 1* is based on a maximum global pooling, *i.e.* the pooling size is equal to the length of the convolution output. In this section, we compare different sizes of pooling window with both the maximum and the average pooling (see Section 1.3.5.3). For each type of pooling, we assess four different models with different window sizes $\in \{global, 10, 100, 400\}$ were global corresponds to the length of the convolution layer

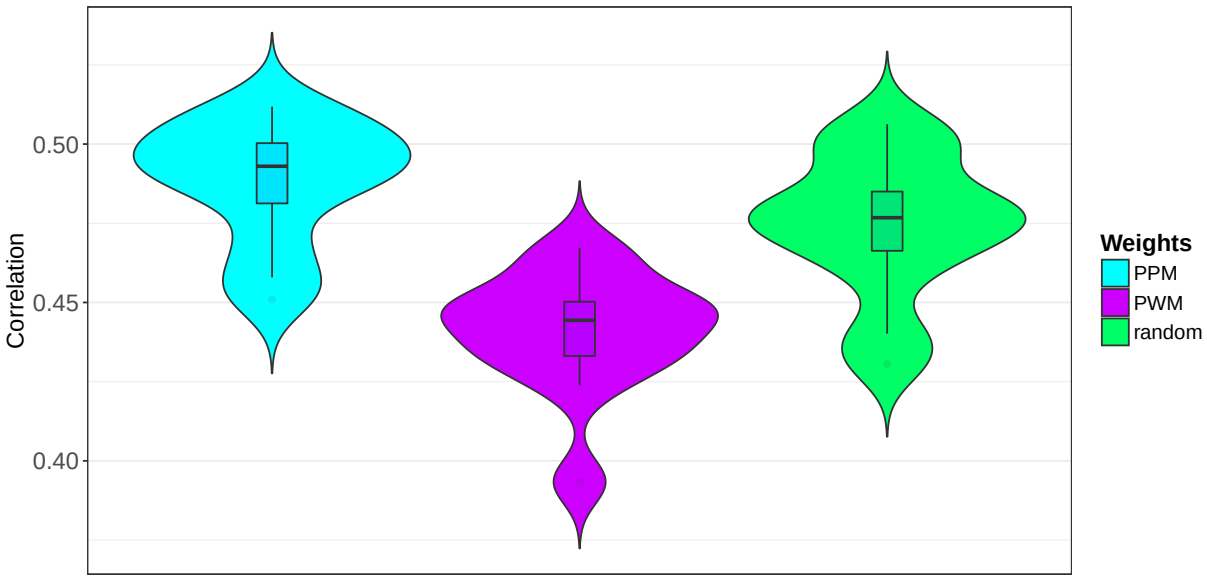


Figure 4.10: ***Comparison of weight initialization in convolution*** *using Spearman correlation on the test set. The first boxplot (cyan) refers to* model 1 *initialized by PPM matrices. The second and third boxplots represent respectively models initialized with PWM matrices and with random values.*

output. Note that the output length of the pooling layer is 1, 2, 9 and 98 for a window size equal to global, 400, 100 and 10 respectively. Figure 4.11 illustrates the results of this parameter tuning. The first boxplot (po_m = global) refers to *model 1* with a global max-pooling. The three other boxplots in cyan refer from the left to the right to networks fitted with maximum pooling with window size = $\{10, 100, 400\}$ respectively. Boxplots in purple illustrates the Spearman correlations of networks fitted using a pooling layer with the average operator with different window size po_av equals to (from left to right) $\{global, 10, 100, 400\}$, where *global* refers to a global average-pooling.

We note that the network trained with a max pooling of window size equal to 10 provides performances slightly lower than an overall max pooling (median correlation = 0.483). However, pooling sizes of 100 and 400 increase model performances with respective median correlations of 0.501 and 0.493. Similar to the maximum operator, an average pooling with a window size equal to 100 provides higher performances that other window sizes. The median correlation using the average are respectively 0.500, 0.465, 0.502 and 0.500

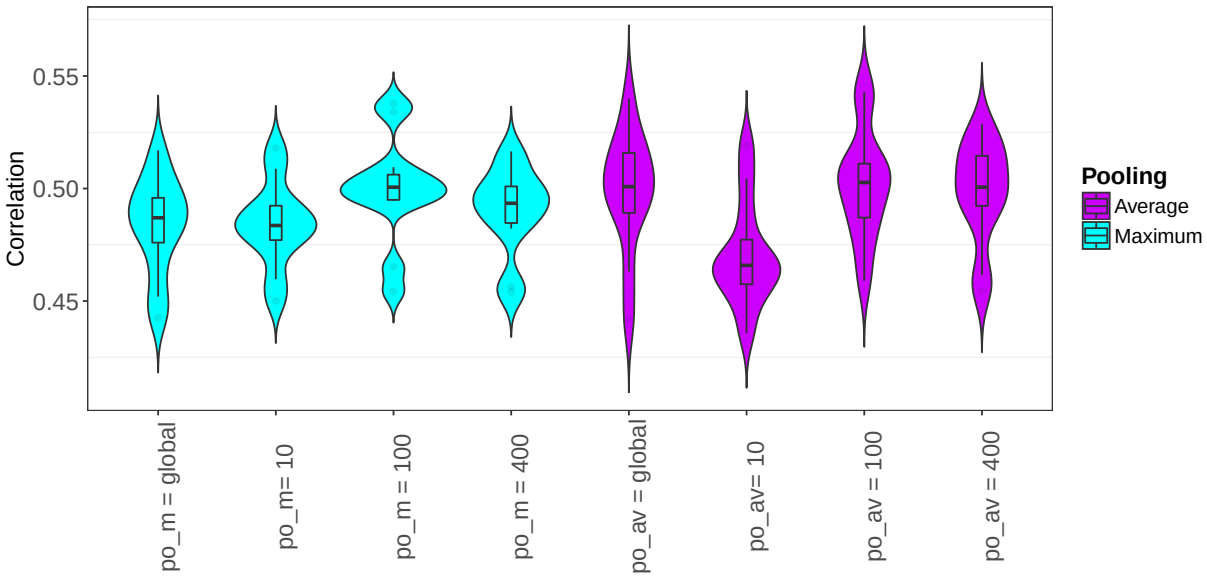


Figure 4.11: ***Comparison of pooling methods and window size*** *using Spearman correlation on the test set. Boxplots in cyan represents the maximum pooling and those in purple represents the average pooling. For each operator (color), each boxplot represents a window size, that is from left to right,* $po = \{global, 10, 100, 400\}$ *where global refers to a pooling over all the output. po_m and po_av means pool for respectively maximum and average. Note that the first boxplot corresponds to* model 1.

for *po_av* equal to global, 10, 100 and 400 respectively. Finally, when comparing the two operators, we can note that there are no significant differences between maximum and average. Furthermore, if we consider the two higher performances, pooling maximum with window equal to 100 and a pooling average with a window equal to 100, median correlation is almost the same with a $p - value$ of Wilcox test of 0.9. Note that the test was applied to 12 tumors.

After this parameter optimization, we define *model 2* as a network with the same parameters as *model 1* but with a maximum pooling layer with a window size equal to 100. This model is used next as a reference model for comparison (median correlation = 0.501).

4.4.2.4 Network without regularization

In this section, we study the importance of the regularization in CNN. For this, we trained a network with the same architecture as *model 2* without the two dropout layer (no regularization). The correlations of this model computed on the test set for 12 tumors are presented in Figure 4.12 (purple boxplot).

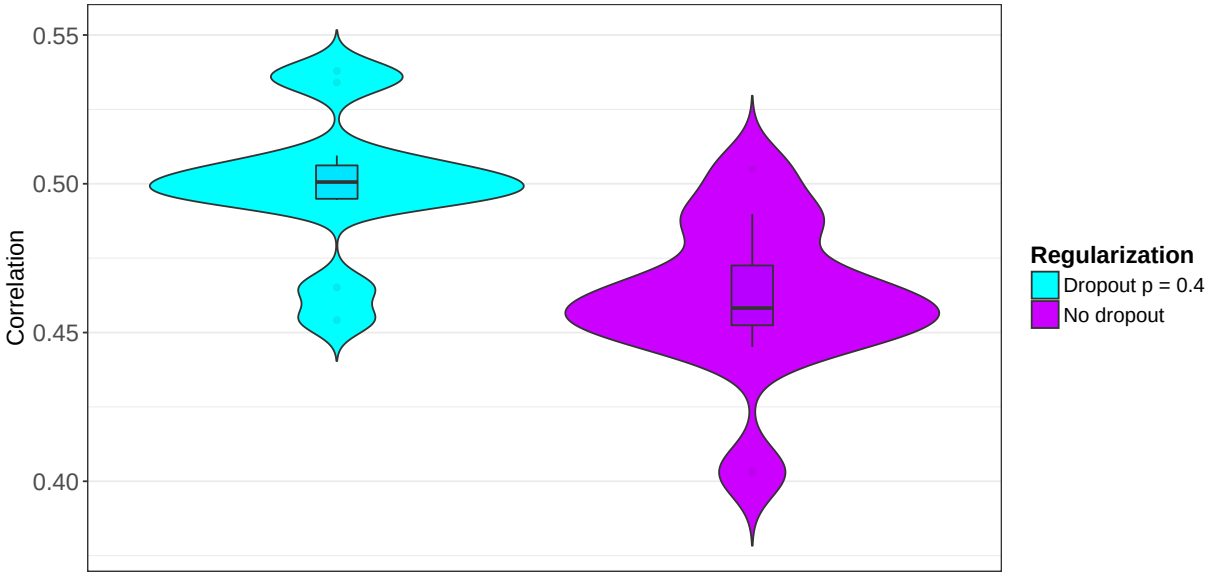


Figure 4.12: ***Network trained with or without dropout regularization*** *The first boxplot (cyan) refers to correlations of* model 2 *where a dropout layers ($p = 0.4$) follows each of the the pooling layer and the ReLU layer. The second boxplot (purple) is correlations of a network trained with no dropout.*

The median correlation is 0.458 versus 0.501 for the *model 2* (cyan boxplot in Figure 4.12) trained with two dropout layers with fraction $p = 0.4$. This shows the importance of dropout and, more generally, of regularization methods when training networks with thousand parameters ($9,937,571$ parameters in our case).

4.4.2.5 Importance of the non-linear dense layer

In Section 1.3.5.4, we explained how a dense layer with ReLU activation function increases non linearity. This layer is characterized by its number of hidden neurons. The model

optimized in previous sections (*model 2* architecture) is fitted with a ReLU dense layer with 2000 neurons. In this section, we compare model performances whit different number of neurons in this layers. For this, we considered two different models with the same architecture as *model 2*, but with changing the number of neurons in the ReLU dense layer of each model. The numbers of neurons considered are $\{200, 400\}$. Besides, we fitted a model without the non-linear dense layer to evaluate the importance of this hidden layer.

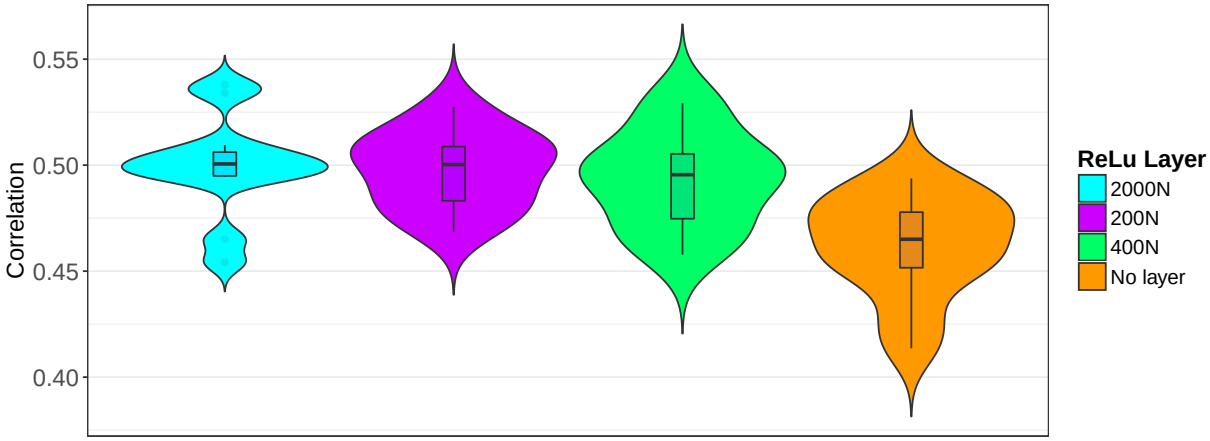


Figure 4.13: ***Optimization of the dense layer following the pooling.*** *The first boxplot (cyan) refers to* model 2 *where a fully connected layer with a ReLU activation function and 2000 neurons follows the pooling layer. Second and third boxplots (purple and green) refer to networks including in their architecture a dense layer with RelU activation function with 200 and 400 neurons respectively. The last boxplot (orange) corresponds to Spearman correlations with a network trained without a non-linear dense layer. Each boxplot represents the Spearman correlations computed for 12 tumors on the test set.*

Figure 4.13 illustrates Spearman correlations of the different models. The first boxplot (cyan) refers to *model 2* architecture with a median correlation of 0.501. First, we notice that the networks fitted with no ReLU dense layers (orange boxplot in Figure 4.13) have the lowest Spearman correlations compared to all tested networks (median correlation = 0.458). This result highlights the importance of the non-linear hidden fully-connected layer in this type of convolution network. Second, the correlations computed by the models fitted with a ReLU layer with 200 (in purple) or 400 neurons (in green) are similar to those of *model 2* with respective median correlations of 0.500 for 200 neurons and 0.498 for 400

neurons. However, the most important difference between these two models is the number of trained parameters: models with 200 neurons and 400 neurons have $1,023,951$ and $2,014,351$ parameters, respectively, which is very lower than that required for *model 2* $(9,937,571)$. This difference induces as well a high difference in execution time. Hence, we define *model 3* as the architecture where the non-linear dense layer with ReLU activation possesses only 200 neurons.

4.4.2.6 Freezing the convolution layer

In this last step, the aim is not to optimize a parameter of the model. Instead, it is to show the importance of updating the weights during convolution. Freezing the weights of the convolution layer means that the backpropagation algorithm does not update the weights of this layer. This method is generally used in transfer learning and pre-trained models [YCBL14], where convolution weights (*i.e.* filters) trained on one network are transferred to another network, instead of training it from scratch. Convolution weights are frozen during the training of the second network, which reduces the number of trained parameters and the execution time, and in some cases, improves network performances [KBFS16].
In our case, we freeze the weights in the first and only one convolution layer, which means that the PPMs will not be updated during the training. All other weights (from dense layers) are updated. This method is applied to *model 3* architecture using the parameter $trainable = FALSE$ in the convolution layer in Keras. Spearman correlations computed on the test set, using the network with frozen convolution weights are presented in Figure 4.14 (green boxplot), with a median correlation of 0.419. Freezing the convolution layer induces a decrease of 9% of correlations compared to *model 3* architecture (median correlation = 0.500). Note that in the network with frozen weights, the number of non-trained parameters is $33,550$ among $1,023,951$ parameter in total. This reduces the training time almost to the half. The last experiment shows that this convolution network does not just learn the possible motifs combinations, but also optimize some motifs and can potentially learn new complementary motifs.

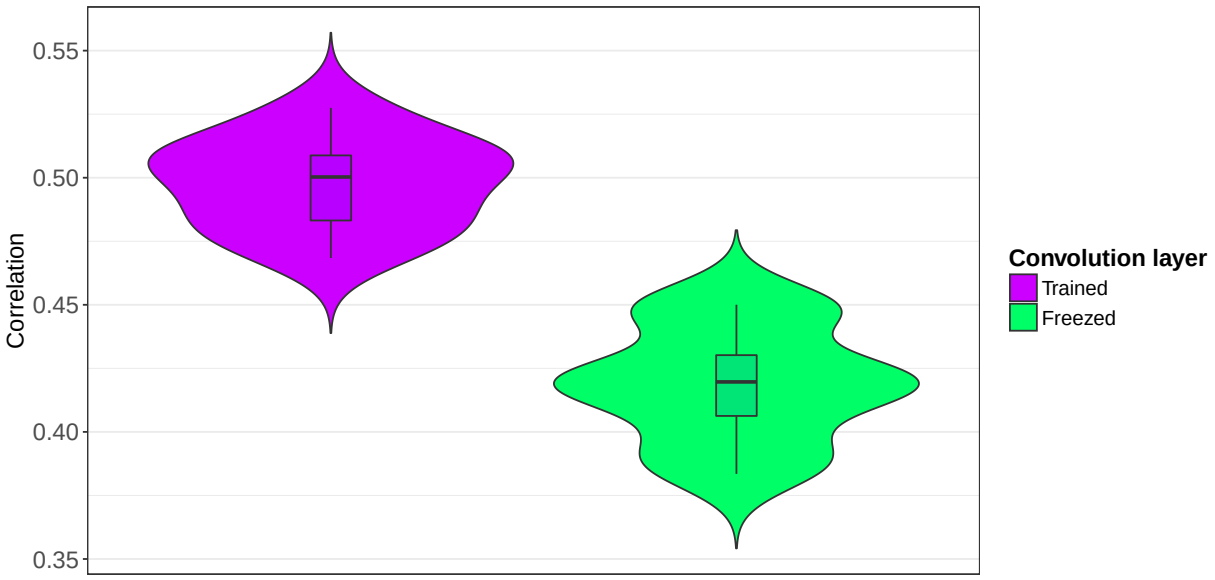


Figure 4.14: ***Performances of networks when freezing the convolution layer.*** *The first boxplot (cyan) refers to* model 3 *where the weights of the convolution layer are trained and updated. The second boxplot (purple) contains Spearman correlations of a network where the convolution layer is frozen and its weights are not updated in the process.*

4.4.3 Optimized architecture

In the previous section, we manually optimized each hyperparameter of the convolution layer independently. The network with the highest median Spearman correlation on the test set was selected. This model, denoted as *model 3*, involves three hidden layers and two dropout layers:

1. Convolution layer with ReLU activation function and 550 filters of size equal to 15. The weights are initialized with position probability matrices (see Section 4.4.1.2).
2. Pooling layer: using the maximum operator and pooling with a window size equal to 100.
3. Dropout layer with fraction $p = 0.4$.
4. Dense layer: with ReLU activation function and 200 neurons.
5. Dropout layer with fraction $p = 0.4$.

A graphical representation of this network is illustrated in Figure 4.15 where for each layer, the optimal parameters found by this study are indicated on the right.

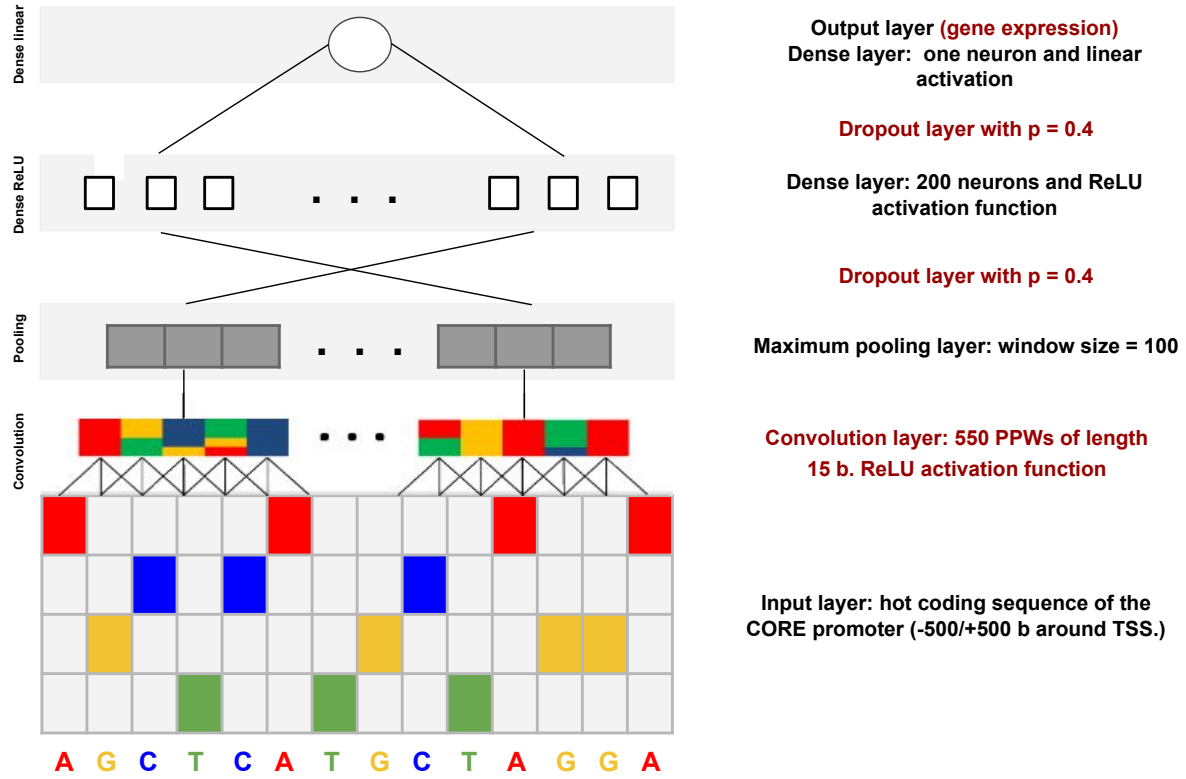


Figure 4.15: ***The optimized convolution neural network architecture.*** *Graphical representation of the convolution network optimized in Section 4.4.2 denoted as* model 3. *For each layer of the network, the parameters of the layer are noted on the right. Figure inspired by [LQX17]*

It is important to note that the architecture retained at this stage was obtained from a set of hyperparameters values. Hence it is not necessarily the most optimized architecture. Notably, the components of the architecture were tested independently and sequentially, and due to lack of time, we did not assess the effect when combining parameters. The point is further discussed in Section 4.5. Furthermore, in this section, we trained a convolution network with only one convolution layer with pre-defined filters. First, one can think of increasing the number of filters by adding filters initialized randomly with the aim to detect novel motifs not known in the literature. Second, we could also think of adding additional convolution layers. In fact, in our network, the first convolution layer detects

the presence of PPMs in the input layer and the output of this layer provides, at each position, the information about the presence/absence of the motif. An additional convolution layer above the PPM level may capture recurrent combinations between different motifs in the same positions or at consecutive positions along the sequence. Note that several studies based on convolution in one-dimension showed that, one their problems, additional convolution layer does not increase model [ZELG16]. However, it is known that neural networks, in general, are dataset specific and no general rules exist. Further work would be to try to add, one by one, hidden convolution layer and evaluate their effect on model performances as well as trying to detect essential variables from each layer based on their estimated weights.

4.4.4 Comparison with Lasso linear regression

In the previous section, we tried to find manually, among different parameters those that best fit a convolution neural network that predicts gene expression using CORE promoter sequences as input. We optimized the architecture noted as *model 3* with a median correlation of 0.500. This model was presented as a potential alternative to the Lasso penalized linear model. To compare the performances of the two models, we fit a Lasso penalized linear model on the training set (17939 genes) using motifs scores calculated on the CORE promoter (see Chapter 2 page 4). We used only motifs scores with length less or equal to 15. The Lasso penalized linear model is then applied and evaluated on 12 tumors using the test set and presents a median correlation of 0.445. The correlations are presented in the second boxplot (blue) in Figure 4.16. The convolution network outperforms the Lasso penalized linear model fitted using only motif scores with an increase of 5.6% in correlations. Finally, we also compared the *model 3* architecture to a Lasso penalized linear model fitted using not only motifs scores but also the nucleotide compositions calculated on the CORE promoter. This last one presents a median correlation of 0.505. Results are presented in the last boxplot of Figure 4.16.

The optimized architecture of the convolution network presents similar results as the linear model fitted using scores of motifs and nucleotide compositions. However, the complexity

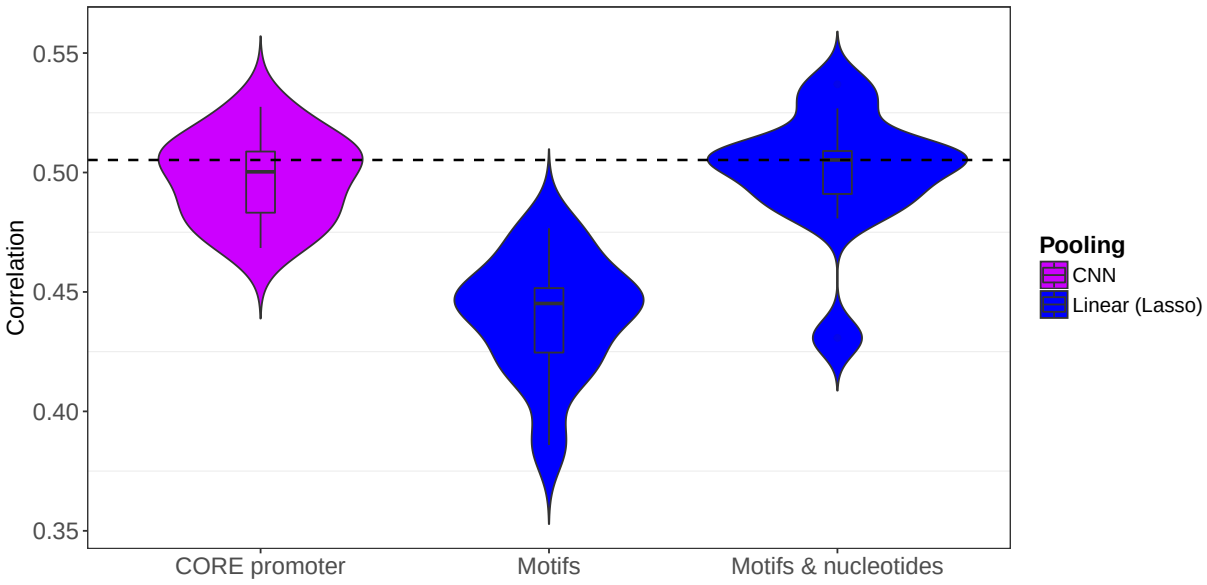


Figure 4.16: ***Comparison between convolution network and Lasso penalized linear regression models.*** *The first boxplot in purple refers to Spearman correlations for the convolution network with the optimized architecture presented in Section 4.4.3 with CORE promoter sequences as input. The boxplots in blue refer to Spearman correlations of the models fitted with ℓ_1 penalized linear regression using: i) scores of motifs in the CORE promoter, and ii) scores of motif and percentages of nucleotides in the CORE promoter. The horizontal line represents the median correlation of the ℓ_1 penalized linear regression fitted with motifs and nucleotides. Spearman correlations are computed for 12 tumors using the test set.*

of the convolution network is much higher than that of the linear model. Furthermore, the Lasso penalization allows us to efficiently identify the most important variables, which renders this approach much more interpretive than convolution networks from a biology perspective (see Chapter 5).

4.5 Discussion about optimization of hyperparameters

In this chapter, for both types of networks (multilayer and convolution), the optimization procedure was manual. For each network, we started with a certain architecture and

then for each parameter independently, we chose a set of potential values and compared networks when changing this values one by one. Even though the final performances were good compared to a Linear model, this is evidently not the best optimization procedure. First, all parameters were not optimized (for example batch size, number of epochs, ...). Second, it is known that some dependencies between different parameters exist and may be very high. For this reason, tuning independently the parameters do not necessarily lead to the best architecture. On the other hand, running a grid-search [PVG+11] algorithm that exhaustively searches through all combinations of hyperparameters suffers from the curse of dimensionality and of prohibitive execution time. Another possibility would be to use a random search strategy such as the one described in reference [BB12]. Random search differs from grid search in that we no longer provide a discrete set of values to explore for each hyperparameter. Rather, we provide a statistical distribution for each hyperparameter from which values may be randomly sampled. Finally, an algorithm that also shown good results in the literature is the Bayesian optimization based on probabilistic models [SLA12]. This algorithm uses the results of the previous iteration to improve the sampling method of the next experiment. The python package ″Hyperopt″ [BYC13] is an implementation of the Bayesian optimization. Future work is to apply these algorithms using our data and networks with the aim to find the best architecture.

Chapter 5

Discussion and perspectives

Gene regulation is the key differentiation between different cell types and functions. Even though the relations between gene expressions and several biological elements (TF, epigenetics, ...) are established ([LLZ14b], [JFLL15b], [SGG+16], [KCL+10], [KSR16]). Models that provide a direct relationship between sequence and gene regulation are very restrained for the moment, especially in the human genome. The main objective of this thesis is to model gene expression based only on genomic variables computed on different regulatory regions of the DNA sequence. For that aim, we proposed two frameworks to model gene expression, one based on a Lasso penalized regression model and the other one based on deep neural networks.

In a first direction, we presented a Lasso penalized linear regression model to predict and explain gene expression based only on features computed from DNA sequences of different regulatory regions. More precisely, our model is based on nucleotide compositions calculated for eight different regulatory regions (promoter, CDS, introns, ...) 2. Surprisingly, we showed that the gene body, especially introns, highly contributed (more than promoters) in predicting gene expression. The accuracy of our model is comparable to models based on experimental data ChIP-seq/DNaseI (RACER [LLZ14b], Rabit [JFLL15b] and TEPIC [SGG+16]). Furthermore, we presented a procedure of predictive variables randomization per gene to evaluate the biological significance and interpretations of each type of predictive variables (sequences features and experimental variables). We showed that

when fitting a model with sequence features randomized by gene, correlations are almost null. This result indicates that our model captures certain combinations between these features. However, this information was not present for ChIP-seq and DNase features (*i.e.* the model stays accurate when randomizing the variables per gene). This result indicates the existence of a bias in experimental data certainly related to the opening of the chromatin. Genes that are present in an open chromatin area are often bound by a high number of TFs, by consequences, scores of ChIP-seq of TFs are almost similar which makes these variables exchangeable in each gene. Very recently, Schmidt & al., the author of TEPIC [SGG+16], discussed the presence of the open chromatin bias in experimental data and, regarding our randomization procedure, presented a method to correct this bias on experimental data based on different score penalizations [SS18]. Even though this approach allows to correct a part of that bias, the results are still not fully satisfying, (*i.e.* correlations after randomization are still not close to zero).However, they observed that the structure of the regression coefficients strongly differ when variables are randomized. With original variables, some coefficients are much more important than others, whereas, after a randomization step, all the coefficients are very similar and close to zero. This result indicates that these experimental data could provide a meaningful interpretation despite the persistent open chromatin bias. Another possible method is to fit a model on randomized experimental predictive variables and then apply this model to the original set of experimental variables (without randomization). If both sets of data provide similar predictions, *i.e.* similar model performances, we can conclude that experimental data have no meaningful interpretation. However, the primary results showed that the performance of the model applied to the original variables present a large decrease compared to those of randomized variables which may be a confirmation for the conclusions of Schmidt & al. [SS18]. This validation process is yet primary, and Chloé Bessiére, a PhD student in the team, is developing additional validations.

Furthermore, we evaluated the contribution of the score of DNA binding motifs in the CORE promoter in predicting gene expression. Even though it is known that the binding of TFs to specific DNA sites (motifs) is essential in regulating gene expression, the contribution of motifs scores in predicting gene expression were not very high compared to those of nucleotide compositions. This result may be due to the definition of the scores

(maximum score when scanning the CORE promoter sequence) based on its probability weight matrix (PWM). Each PWM provides a huge number of binding sites (motifs) while it was shown that only a few of these sites are bound by TFs [KLS+11]. To overcome the limitations of certain PWMs, a method was proposed in our team to provide a binary classifier to provide a binary classifier approach of the presence of a TF binding site in a region based on TF combinations [VCL+17]. The prediction obtained by this approach can be used as predictive variables (instead of PWM scores) in our model to predict gene expression. A perspective would be to evaluate their contributions to our model. Note that we will further highlight the importance of the motifs by using their position probability matrix (PPM) as filters in convolution networks.

Moreover, in Chapter 2 we briefly presented an attempt to detect mutational signatures based on stable nucleotide compositions in different regulatory regions. Our method is based on the mutation signatures provided in [LSP+13]. These mutations were computed using only exome genome sequencing which is considered as a limitation of the method. Besides, our model was biased by the CpG content that is at a time highly mutated and with high importance for gene expression prediction. Actually, characterization of mutational signatures for regulatory regions is being addressed. In 2016 and in 2018, two papers ([KTS16] & [CGB+18]) were published with the aim of providing mutation profiles of different TFBSs (motifs). These methods are advantageous as they are based on whole genome mutation annotations and take non-coding sequences into consideration. On one hand, Kaiser & al. [KTS16] concentrated on characterizing each motif by a mutational profile across different cancer types. They considered two types of alterations (mutations) those that transform a binding motif to a non-binding (disruption) and vice-versa (creation). They succeeded to identify alterations signatures for 81 motifs in 11 cancer types, and they showed that functional TFBSs (motifs) are enriched for mutations in the different cancer type. Their method was based on a comparison of PWM-scores between functional altered motifs and a defined reference motif for each single base substitution occurring within tri-nucleotides. On the other hand, Chan & al. [CGB+18] highlighted the impact of mutations on motifs in cancer and also constructed mutational signatures for 512 TFBS (motifs) in different cancer types. They proceed with a Bayesian framework and compute each motif signature in function of the probability of motif alteration conditionally to the

occurred mutation (m_k) and the probability of this mutation conditionally to the existing signatures. An alteration can be a creation or a disruption, and the possible mutation (m_k) corresponds to tri-nucleotide alteration. The methods proposed in both articles may provide a possible solution to overcome the limitations of the CpG in our proposed approach. On a different direction, and with the aim of refining the model presented in Chapter 2, Christophe Menichelli, a PhD student in the team, developed a new method recently denoted Dexter. This approach can detect from the DNA sequence, possible k-mers from different regulatory regions that are important in gene expression prediction. This method may be used to further investigate signature mutations on different k-mers in different regulatory regions based on whole sequencing mutation counts.

Finding sequence-level instructions in DNA is an emerging field. As said in Chapter 3, recently, three papers were published to predict gene expression based on DNA sequences using artificial neural networks ([ZTY+18], [AS18], [KRB+18]). In this paragraph, I will discuss the global approach as well as some results and limitations. The details of the convolution architecture will be discussed in the following paragraph.
First, Zhou & al. [ZTY+18] developed a two-stage model, denoted ExPecto, to predict gene expression based on DNA sequences and epigenetic marks. The first stage is an extension of their previous model, Deepsea [ZT15] to predict epigenetic variants based on DNA sequences using a deep convolution network. The second stage is based on a Ridge (ℓ_2) penalized linear regression to predict gene expression based on the predictions of the epigenetics in the first stage of the model. This strategy yielded a median Spearman correlation, between observed and predicted gene expression, of 0.81 over 218 tissues. Even though their aim is to predict gene expression based on the DNA sequences, they use experimental data as well as an intermediate predictive variable which provides a higher order of information from both sequence and experimental variables. However, the comparison with our model performances is not evident as our model is based only on the information containing in the sequence.
The Second paper is a work of Agarwal & Shendure [AS18] posted in bioRxiv. They aim at predicting gene expression based only on the sequence and on sequence features related to mRNA half-life (stability and degradation) using a convolution network. Different models

were trained and their retained model (with the highest accuracy) was built to predict the median gene expression across 56 tissues using the DNA sequence located at -7000/+3500 b around the TSS as well as the CG content and the lengths of different functional regions (UTR, exon, introns, ...). Their model presented high accuracy in human and mouse respectively. The highest accuracy was obtained when considering a region of -7000/+3500 around TSS. However, they showed that the majority of the information was captured within smaller regions localized around the TSS. Furthermore, the performances of their model when used to predict gene expression in a cell type-specific were lower than those when predicting the median of expression over 56 tissues. However, their choice of resuming the information of gene expression over the tissues in the median value is not arbitrary. For that they showed that their set of gene expression (based only on coding genes) is highly correlated among pairs of tissues (median gene expression correlation among 56 tissues of 73%). Indeed, coding gene expressions are highly correlated among different tissues as well as among cancer tissues. This high correlation presents a limitation to condition-specific models based on coding genes when it comes to differentiating cancer types (see Chapter 3). This assumption is validated by a model swap approach. For that, we fitted a model on one tumor from a cancer type, then applied the model and computed its accuracy on different tumors from different cancer types. Surprisingly, the correlations were not very different from a cancer type to another. This result means that our model is not able to differentiate different cancer types and one potential cause is the high correlations between coding genes among different cancer types. However, this limitation almost faded away when including non-coding genes. The correlations between genes in different cancer types when considering both coding and non-coding gene expression decrease vastly (almost of 30% compared to only considering coding genes). A potential perspective will be to adjust a statistical model to predict gene expression (coding and non-coding) that can discriminate the different types of genes. Such a model may provide a significant differentiation between cell-types.

The third and final paper is published by Kelly & al. [KRB+18] in spring 2018. Their article is not restrained to modeling gene expression. They present an approach of convolution neural network, to predict different types of variables, in particular CAGE genes expression, based only on DNA sequences. Their method denoted Basenji is a modified version of

their previous model [KSR16] that they presented to predict peaks using DNA sequence (classification model). The median Pearson correlation, between observed and predicted CAGE gene expression, across different cell types, is 0.86. Besides, they showed that the performances of Bessenji are higher than their previous model (Basset). Note that their study was not restrained to the promoters. Actually, they extract 131 KB sequences across the chromosomes. Several other biological aspects were also addressed in this paper especially about the influence of distal regions on gene expression.

Note that these last two approaches ([AS18], [KRB+18]) present very high performances based on DNA sequence, even higher than models fitted based only on experimental data. Further validations of the methods and the dataset is necessary to understand these results

0

With the aim of increasing the performances of our model, we considered interactions. The concept of biological interactions itself inspired the idea of adding interactions between different nucleotide compositions. In Chapter 1 we presented the different interactions existing in biology and controlling gene regulation. The approach that we presented induced a slight increase in model performances with selections of some significant interactions between different regions. In our approach, we used mathematical operations to define an interaction (minimum, maximum and product). However, this may not be the best way to define an interaction. Using the maximum or the minimum definition may create new interactions variables that are very correlated to original variables. This limitation is further highlighted by the approach of stability where we showed that when using only interaction variables to fit a Lasso penalized linear regression, the performances were high, but no variables were stable. Besides, the interpretation of interactions defined as the product is difficult to interpret in biology. Furthermore, we noticed that all nucleotide compositions are significant in the model *i.e.* all $p - values$ of original test (Section 3.2.1) are very low. Such results complicate the rules to select an interaction. To overcome this limitation, we can define different selection rules specific for each interaction, instead of a global threshold rule. One rule that we aim to consider is to select an interaction between two variables if it is more important to the model that both original variables. Another limitation of this method is its high computational cost even when considering only second-order interactions which makes it difficult to consider higher order interactions. All these limitations

and other causes motivated the idea of our last project based on neural networks. As I will explain in the following, via neural networks, we can detect interactions between variables at an even higher level than second-order.
Finally, for each approach presented with the aim to detect interactions, further validation should be applied, if possible, by comparing the selected interactions to those already existing in the biological literature. For instance, a potential validation track is Bunting & al [BSS+16]. They presented biological interactions within promoters and emphasis on the existence of 5' to 3' gene looping that may induce interactions with these regions.

The second framework we presented to predict gene expression consists of two types of artificial neural networks: deep networks and convolution networks. The hyperparameters of each neural network were optimized based on a manual search. We discussed in Chapter 4 the importance of using algorithms to optimize parameters considering the dependencies between them. In [AS18], Agarwal & al. used the "hyperopt" package from python that to apply the random search algorithm to optimize different parameters. Also, they compared the optimized parameters by this algorithm to manual search optimizations and showed that the model optimized by a random search outperformed the network based on a manual optimization with a difference of almost 7% of accuracy. On another hand, Kelley & al. [BSS+16] optimized the hyperparameters of Basenji based on a Bayesian optimization algorithm. A first important direction is to apply one or both algorithms to our network and optimize architectures (for MLP and CNN), compare the different optimization methods and retain the high accuracy model.
In this project, we examined the performances of both multilayer networks and convolutions networks to performances of a Lasso penalized regression model. The accuracies of the models were very similar in term of Spearman correlations. However, the Lasso penalized regression model is much simpler to explain and to extract significant features with substantial effects on gene expression while neural networks are more like a black box. Nonetheless, recent studies are being developed by the Anshul Kundaje team (https://sites.google.com/site/anshulkundaje/Home) to extract some critical information from neural networks hidden layers. In 2017, they presented DeepLIFT for Deep Learning Important FeaTures [SGK17]. In their procedure, first, they applied a neural

network and learned the weights. Further, they assigned a score of contribution for each neuron considering the difference between its activation and a reference activation defined in prior. Computing the scores also depends on the sign of the difference (this overcomes the limitation of saturation). These scores can describe the most important features to predict the response variable. Besides, in the paper, they present different strategies to define the reference depending on the dataset and the dimension of the input. At the beginning of September 2018, researchers from the same team published their latest study denoted DFIM for Deep Features Interactions Maps [GSFK18]. DFIM is based on convolution networks to detect synergistic interactions between DNA features, especially nucleotide compositions and motifs. For each interaction, they computed a score using the partial derivation of the output depending on the input (gradient of the network). One of our perspectives is to apply these methods to our convolution network and detect essential features (nucleotide compositions and motifs in the sequences) as well as potential interactions between motifs in the CORE promoter. However, to be able to detect inter and intra regions interactions, we should adapt our convolution network with a larger sequence in input to include more regulatory regions (UTR, gene body, ...).

Finally, our convolution network does not take into consideration the nucleotide compositions of the DNA sequences. The performances of our network with one convolution layer and PPM as filters are similar to those of a Lasso penalized linear regression fitted to predict gene expression based on motifs scores and nucleotide compositions. This result highlights the importance of motifs provide the fact that the results suggested that the PPMs may cover the effect of nucleotide compositions Another exciting direction could be to add nucleotide compositions from different regulatory regions as neurons in the latest dense layers in the convolution networks. This addition may establish combinations between motifs and nucleotide compositions that may increase the accuracy of predicting gene expression. This idea was inspired by [AS18] who showed that an increase of about 9% occurs on model performances when adding continuous variables as neurons in their first dense layers.

Bibliography

[A+97] John Aldrich et al. Ra fisher and the making of maximum likelihood 1912-1922. *Statistical science*, 12(3):162–176, 1997.

[ADWF15] Babak Alipanahi, Andrew Delong, Matthew T Weirauch, and Brendan J Frey. Predicting the sequence specificities of dna-and rna-binding proteins by deep learning. *Nature biotechnology*, 33(8):831, 2015.

[Ait36] Alexander C Aitken. Iv.âĂŤon least squares and linear combination of observations. *Proceedings of the Royal Society of Edinburgh*, 55:42–48, 1936.

[ALRS17] Christof Angermueller, Heather J Lee, Wolf Reik, and Oliver Stegle. Deepcpg: accurate prediction of single-cell dna methylation states using deep learning. *Genome biology*, 18(1):67, 2017.

[And15] Robin Andersson. Promoter or enhancer, what's the difference? deconstruction of established distinctions and presentation of a unifying model. *Bioessays*, 37(3):314–323, 2015.

[Ann08] A Annunziato. Dna packaging: nucleosomes and chromatin. *Nature Education*, 1(1):26, 2008.

[ANZW+13] Ludmil B Alexandrov, Serena Nik-Zainal, David C Wedge, Samuel AJR Aparicio, Sam Behjati, Andrew V Biankin, Graham R Bignell, Niccolo Bolli, Ake Borg, Anne-Lise Børresen-Dale, et al. Signatures of mutational processes in human cancer. *Nature*, 500(7463):415, 2013.

[AOA06] Sigrid D Auweter, Florian C Oberstrass, and Frederic H-T Allain. Sequence-specific binding of single-stranded rna: is there a code for recognition? *Nucleic acids research*, 34(17):4943–4959, 2006.

[AS95] Christopher H Achen and W Phillips Shively. *Cross-level inference*. University of Chicago Press, 1995.

[AS18] Vikram Agarwal and Jay Shendure. Predicting mrna abundance directly from genomic sequence using deep convolutional neural networks. *bioRxiv*, page 416685, 2018.

[AWLS16] Silvio Amir, Byron C Wallace, Hao Lyu, and Paula Carvalho Mário J Silva. Modelling context with user embeddings for sarcasm detection in social media. *arXiv preprint arXiv:1607.00976*, 2016.

[BB12] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(Feb):281–305, 2012.

[Ben12] Yoshua Bengio. Practical recommendations for gradient-based training of deep architectures. In *Neural networks: Tricks of the trade*, pages 437–478. Springer, 2012.

[BF15] Deepak Babu and Melissa J Fullwood. 3d genome organization in health and disease: emerging opportunities in cancer translational medicine. *Nucleus*, 6(5):382–393, 2015.

[BFOS84] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Wadsworth and Brooks, Monterey, CA, 1984.

[BGV92] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152. ACM, 1992.

[BH95] Yoav Benjamini and Yosef Hochberg. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the royal statistical society. Series B (Methodological)*, pages 289–300, 1995.

[BKBY18] Sumanta Basu, Karl Kumbier, James B Brown, and Bin Yu. iterative random forests to discover predictive and stable high-order interactions. *Proceedings of the National Academy of Sciences*, page 201711236, 2018.

[BMS77] Susan M Berget, Claire Moore, and Phillip A Sharp. Spliced segments at the 5âĂš terminus of adenovirus 2 late mrna. *Proceedings of the National Academy of Sciences*, 74(8):3171–3175, 1977.

[Bre96] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[Bre01] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[BSS+16] Karen L Bunting, T David Soong, Rajat Singh, Yanwen Jiang, Wendy Béguelin, David W Poloway, Brandon L Swed, Katerina Hatzi, William Reisacher, Matt Teater, et al. Multi-tiered reorganization of the genome during b cell affinity maturation anchored by a germinal center-specific locus control region. *Immunity*, 45(3):497–512, 2016.

[BT04] Michael A Beer and Saeed Tavazoie. Predicting gene expression from sequence. *Cell*, 117(2):185–198, 2004.

[BYC13] James Bergstra, Dan Yamins, and David D Cox. Hyperopt: A python library for optimizing the hyperparameters of machine learning algorithms. In *Proceedings of the 12th Python in Science Conference*, pages 13–20. Citeseer, 2013.

[C+15] François Chollet et al. Keras. https://keras.io, 2015.

[CAM+12] Chao Cheng, Roger Alexander, Renqiang Min, Jing Leng, Kevin Y Yip, Joel Rozowsky, Koon-Kiu Yan, Xianjun Dong, Sarah Djebali, Yijun Ruan, et al. Understanding transcriptional regulation by integrative analysis of transcription factor binding data. *Genome research*, 22(9):1658–1667, 2012.

[Cau47] Augustin Cauchy. Méthode générale pour la résolution des systemes dâĂŹéquations simultanées. *Comp. Rend. Sci. Paris*, 25(1847):536–538, 1847.

[CC12] Michal Chorev and Liran Carmel. The function of introns. *Frontiers in genetics*, 3, 2012.

[CGB+18] Calvin WY Chan, Zuguang Gu, Matthias Bieg, Roland Eils, and Carl Herrmann. Impact of cancer mutational signatures on transcription factor motifs in the human genome. *bioRxiv*, page 422386, 2018.

[CGBR77] Louise T Chow, Richard E Gelinas, Thomas R Broker, and Richard J Roberts. An amazing sequence arrangement at the 5âĂš ends of adenovirus 2 messenger rna. *Cell*, 12(1):1–8, 1977.

[Cle79] William S Cleveland. Robust locally weighted regression and smoothing scatterplots. *Journal of the American statistical association*, 74(368):829–836, 1979.

[Clo09] ClockBackward. Ordinary least squares linear regression: Flaws, problems and pitfalls. http://www.clockbackward.com/2009/06/18/ordinary-least-squares-linear-regression-flaws-problems-and-pitfalls/, 2009.

[com16] Computational pan-genomics: status, promises and challenges. *Briefings in Bioinformatics*, 19(1):118–135, 2016.

[Cox87] Christopher Cox. Threshold dose-response models in toxicology. *Biometrics*, pages 511–523, 1987.

[CSMB81] Gérard CUNY, Philippe SORIANO, Gabriel MACAYA, and Giorgio BERNARDI. The major components of the mouse and human genomes: 1. preparation, basic properties and compositional heterogeneity. *European Journal of Biochemistry*, 115(2):227–233, 1981.

[CV95] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[CW00] Avril Coghlan and Kenneth H Wolfe. Relationship of codon bias to mrna concentration and protein length in saccharomyces cerevisiae. *Yeast*, 16(12):1131–1145, 2000.

[DB11] Aimée M Deaton and Adrian Bird. Cpg islands and the regulation of transcription. *Genes & development*, 25(10):1010–1022, 2011.

[DBZ04] Debopriya Das, Nilanjana Banerjee, and Michael Q Zhang. Interacting models of cooperative gene regulation. *Proceedings of the National Academy of Sciences of the United States of America*, 101(46):16234–16239, 2004.

[DGR16] Jesse R Dixon, David U Gorkin, and Bing Ren. Chromatin domains: the unit of chromosome organization. *Molecular cell*, 62(5):668–680, 2016.

[dL94] M Ponz de Leon. Oncogenes and tumor suppressor genes. In *Familial and Hereditary Tumors*, pages 35–47. Springer, 1994.

[Dra03] Sorin Draghici. *Data analysis tools for DNA microarrays*. Chapman and Hall/CRC, 2003.

[DS18] Lan TM Dao and Salvatore Spicuglia. Transcriptional regulation by promoters with enhancer function. *Transcription*, pages 1–8, 2018.

[DSY+12] Jesse R Dixon, Siddarth Selvaraj, Feng Yue, Audrey Kim, Yan Li, Yin Shen, Ming Hu, Jun S Liu, and Bing Ren. Topological domains in mammalian genomes identified by analysis of chromatin interactions. *Nature*, 485(7398):376, 2012.

[EF76] James E Ertel and Edward B Fowlkes. Some algorithms for linear spline and piecewise multiple linear regression. *Journal of the American Statistical Association*, 71(355):640–648, 1976.

[Est08] Manel Esteller. Epigenetics in cancer. *New England Journal of Medicine*, 358(11):1148–1159, 2008.

[FB17] Valeria Fonti and Eduard Belitser. Feature selection using lasso. *VU Amsterdam Research Paper in Business Analytics*, 2017.

[FG53] Rosalind E Franklin and Raymond G Gosling. Molecular configuration in sodium thymonucleate. *Nature*, 171:740–741, 1953.

[FHT10] Jerome Friedman, Trevor Hastie, and Rob Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of statistical software*, 33(1):1, 2010.

[FK03] Nir Friedman and Daphne Koller. Being bayesian about network structure. a bayesian approach to structure discovery in bayesian networks. *Machine learning*, 50(1-2):95–125, 2003.

[FLP+09] Melissa J Fullwood, Mei Hui Liu, You Fu Pan, Jun Liu, Han Xu, Yusoff Bin Mohamed, Yuriy L Orlov, Stoyan Velkov, Andrea Ho, Poh Huay Mei, et al. An oestrogen-receptor-α-bound human chromatin interactome. *Nature*, 462(7269):58, 2009.

[Fri91] Jerome H Friedman. Multivariate adaptive regression splines. *The annals of statistics*, pages 1–67, 1991.

[FV83] Andrew P Feinberg and Bert Vogelstein. Hypomethylation distinguishes genes of some human cancers from their normal counterparts. *Nature*, 301(5895):89, 1983.

[Gal86] Francis Galton. Regression towards mediocrity in hereditary stature. *The Journal of the Anthropological Institute of Great Britain and Ireland*, 15:246–263, 1886.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[GD13] Johan H Gibcus and Job Dekker. The hierarchy of the 3d genome. *Molecular cell*, 49(5):773–782, 2013.

[GHT14] Stefanie Gerstberger, Markus Hafner, and Thomas Tuschl. A census of human rna-binding proteins. *Nature Reviews Genetics*, 15(12):829, 2014.

[GSFK18] Peyton Greenside, Tyler Shimko, Polly Fordyce, and Anshul Kundaje. Discovering epistatic feature interactions from neural network models of regulatory dna sequences. *Bioinformatics*, 34(17):i629–i637, 2018.

[HCN76] Victor Hasselblad, John P Creason, and William C Nelson. Regression using hockey stick functions. 1976.

[Her93] Nouria Hernandez. Tbp, a universal eukaryotic transcription factor? *Genes Dev*, 7(7B):1291–308, 1993.

[HFG+12] Jennifer Harrow, Adam Frankish, Jose M Gonzalez, Electra Tapanari, Mark Diekhans, Felix Kokocinski, Bronwen L Aken, Daniel Barrell, Amonida Zadissa, Stephen Searle, et al. Gencode: the reference human genome annotation for the encode project. *Genome research*, 22(9):1760–1774, 2012.

[HK70] Arthur E Hoerl and Robert W Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

[Hoc76] Ronald R Hocking. A biometrics invited paper. the analysis and selection of variables in linear regression. *Biometrics*, 32(1):1–49, 1976.

[HRH+17] Chung-Chau Hon, Jordan A Ramilowski, Jayson Harshbarger, Nicolas Bertin, Owen JL Rackham, Julian Gough, Elena Denisenko, Sebastian Schmeier, Thomas M Poulsen, Jessica Severin, et al. An atlas of human long non-coding rnas with accurate 5âĂš ends. *Nature*, 543(7644):199, 2017.

[HSS] Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. Lecture 6a overview of mini-batch gradient descent (2012). *Coursera Lecture slides https://class. coursera. org/neuralnets-2012-001/lecture*.

[HTF01] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. Springer series in statistics. *The elements of statistical learning: data mining, inference, and prediction*, 2001.

[HW63] David H Hubel and TN Wiesel. Shape and arrangement of columns in cat's striate cortex. *The Journal of physiology*, 165(3):559–568, 1963.

[IGSDS+15] Robert S Illingworth, Ulrike Gruenewald-Schneider, Dina De Sousa, Shaun Webb, Cara Merusi, Alastair RW Kerr, Keith D James, Colin Smith, Robert Walker, Robert Andrews, et al. Inter-individual variability contrasts with regional homogeneity in the human brain dna methylome. *Nucleic acids research*, 43(2):732–744, 2015.

[Jai17] Rishabh Kumar Jain. Rajeswari ponnuru (intel), ajit kumar p.(intel), and ravi keron n.(intel). cifar-10 classification using intel optimization for tensorflow, 2017.

[JB17] Kamel Jabbari and Giorgio Bernardi. An isochore framework underlies chromatin architecture. *PloS one*, 12(1):e0168023, 2017.

[JBG03] Ronald Jansen, Harmen J Bussemaker, and Mark Gerstein. Revisiting the codon adaptation index from a whole-genome perspective: analyzing the relationship between gene expression and codon occurrence in yeast using a variety of models. *Nucleic acids research*, 31(8):2242–2251, 2003.

[JFLL15a] Peng Jiang, Matthew L Freedman, Jun S Liu, and Xiaole Shirley Liu. Inference of transcriptional regulation in cancers. *Proceedings of the National Academy of Sciences*, page 201424272, 2015.

[JFLL15b] Peng Jiang, Matthew L Freedman, Jun S Liu, and Xiaole Shirley Liu. Inference of transcriptional regulation in cancers. *Proceedings of the National Academy of Sciences*, 112(25):7731–7736, 2015.

[JMMW07] David S Johnson, Ali Mortazavi, Richard M Myers, and Barbara Wold. Genome-wide mapping of in vivo protein-dna interactions. *Science*, 316(5830):1497–1502, 2007.

[Joh09] Wilhelm Johannsen. *Elemente der Erblichkeitslehre*. Fischer, 1909.

[JWPP00] Robert E Johnson, M Todd Washington, Satya Prakash, and Louise Prakash. Fidelity of human dna polymerase η. *Journal of Biological Chemistry*, 275(11):7447–7450, 2000.

[KA99] Min-Hao Kuo and C David Allis. In vivo cross-linking and immunoprecipitation for studying dynamic protein: Dna associations in a chromatin environment. *Methods*, 19(3):425–433, 1999.

[KB14] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[KBFS16] Maarten C Kruithof, Henri Bouma, Noëlle M Fischer, and Klamer Schutte. Object recognition using deep convolutional neural networks with complete transfer and partial frozen layers. In *Optics and Photonics for Counterterrorism, Crime Fighting, and Defence XII*, volume 9995, page 99950K. International Society for Optics and Photonics, 2016.

[KCL+10] Rosa Karlić, Ho-Ryun Chung, Julia Lasserre, Kristian Vlahoviček, and Martin Vingron. Histone modification levels are predictive for gene expression. *Proceedings of the National Academy of Sciences*, 107(7):2926–2931, 2010.

[KCLE81] Michael A Keene, Victor Corces, Ky Lowenhaupt, and SC Elgin. Dnase i hypersensitive sites in drosophila chromatin occur at the 5'ends of regions of transcription. *Proceedings of the National Academy of Sciences*, 78(1):143–146, 1981.

[KFS+17] Aziz Khan, Oriol Fornes, Arnaud Stigliani, Marius Gheorghe, Jaime A Castro-Mondragon, Robin van der Lee, Adrien Bessy, Jeanne Cheneby, Shubhada R Kulkarni, Ge Tan, et al. Jaspar 2018: update of the open-access database of transcription factor binding profiles and its web framework. *Nucleic acids research*, 46(D1):D260–D266, 2017.

[KKPG+13] Maya Kasowski, Sofia Kyriazopoulou-Panagiotopoulou, Fabian Grubert, Judith B Zaugg, Anshul Kundaje, Yuling Liu, Alan P Boyle, Qiangfeng Cliff Zhang, Fouad Zakharia, Damek V Spacek, et al. Extensive variation in chromatin states across humans. *Science*, 342(6159):750–752, 2013.

[KLS+11] Tommy Kaplan, Xiao-Yong Li, Peter J Sabo, Sean Thomas, John A Stamatoyannopoulos, Mark D Biggin, and Michael B Eisen. Quantitative models of the mechanisms that control genome-wide patterns of transcription factor binding during early drosophila development. *PLoS genetics*, 7(2):e1001290, 2011.

[KMC98] Samuel Karlin, Jan Mrázek, and Allan M Campbell. Codon usages in different gene classes of the escherichia coli genome. *Molecular microbiology*, 29(6):1341–1355, 1998.

[Koo12] Eugene V Koonin. Does the central dogma still stand? *Biology direct*, 7(1):27, 2012.

[KPEF15] Roshan Karki, Deep Pandya, Robert C Elston, and Cristiano Ferlini. Defining âĂIJmutationâĂİ and âĂIJpolymorphismâĂİ in the era of personal genomics. *BMC medical genomics*, 8(1):37, 2015.

[Kra96] Stefan Kramer. Structural regression trees. In *AAAI/IAAI, Vol. 1*, pages 812–819. Citeseer, 1996.

[KRB+18] David R Kelley, Yakir Reshef, Maxwell Bileschi, David Belanger, Cory Y McLean, and Jasper Snoek. Sequential regulatory activity prediction across chromosomes with convolutional neural networks. *Genome research*, pages gr–227819, 2018.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[KSR16] David R Kelley, Jasper Snoek, and John L Rinn. Basset: learning the regulatory code of the accessible genome with deep convolutional neural networks. *Genome research*, 2016.

[KTS16] Vera B Kaiser, Martin S Taylor, and Colin A Semple. Mutational biases drive elevated rates of substitution at regulatory sites across cancer types. *PLoS genetics*, 12(8):e1006207, 2016.

[KWG+13] Helena Kilpinen, Sebastian M Waszak, Andreas R Gschwind, Sunil K Raghav, Robert M Witwicki, Andrea Orioli, Eugenia Migliavacca, Michaël Wiederkehr, Maria Gutierrez-Arcelus, Nikolaos I Panousis, et al. Coordinated effects of sequence variation on dna binding, chromatin structure, and transcription. *Science*, 2013.

[L+16] Y LeCun et al. Lenet-5, convolutional neural networks (2015). *Retrieved June*, 1, 2016.

[LAVBW+09] Erez Lieberman-Aiden, Nynke L Van Berkum, Louise Williams, Maxim Imakaev, Tobias Ragoczy, Agnes Telling, Ido Amit, Bryan R Lajoie, Peter J Sabo, Michael O Dorschner, et al. Comprehensive mapping of long-range interactions reveals folding principles of the human genome. *science*, 326(5950):289–293, 2009.

[LBS+07] Liana F Lareau, Angela N Brooks, David AW Soergel, Qi Meng, and Steven E Brenner. The coupling of alternative splicing and nonsense-mediated mrna decay. 2007.

[Leg05] Adrien Marie Legendre. *Nouvelles méthodes pour la détermination des orbites des comètes*. F. Didot, 1805.

[LJZ15] Liang Liu, Guangxu Jin, and Xiaobo Zhou. Modeling the relationship of epigenetic modifications to transcription factor binding. *Nucleic acids research*, 43(8):3873–3885, 2015.

[LLZ14a] Yue Li, Minggao Liang, and Zhaolei Zhang. Regression analysis of combined gene expression regulation in acute myeloid leukemia. *PLoS computational biology*, 10(10):e1003908, 2014.

[LLZ14b] Yue Li, Minggao Liang, and Zhaolei Zhang. Regression analysis of combined gene expression regulation in acute myeloid leukemia. *PLoS Comput Biol*, 10(10):e1003908, 2014.

[LM15] Michael Lynch and Georgi K. Marinov. The bioenergetic costs of a gene. *Proceedings of the National Academy of Sciences*, 2015.

[LML+13] Chaochun Liu, Bibekanand Mallick, Dang Long, William A Rennie, Adam Wolenc, C Steven Carmack, and Ye Ding. Clip-based prediction of mammalian microrna binding sites. *Nucleic acids research*, 41(14):e138–e138, 2013.

[Loh02] Wei-Yin Loh. Regression tress with unbiased variable selection and interaction detection. *Statistica Sinica*, pages 361–386, 2002.

[LQLM10] Xiao Li, Gerald Quon, Howard D Lipshitz, and Quaid Morris. Predicting in vivo binding sites of rna-binding proteins using mrna secondary structure. *Rna*, 2010.

[LQX17] Yi Li, Daniel Quang, and Xiaohui Xie. Understanding sequence conservation with deep learning. In *Proceedings of the 8th ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics*, pages 400–406. ACM, 2017.

[LSP+13] Michael S Lawrence, Petar Stojanov, Paz Polak, Gregory V Kryukov, Kristian Cibulskis, Andrey Sivachenko, Scott L Carter, Chip Stewart, Craig H Mermel, Steven A Roberts, et al. Mutational heterogeneity in cancer and the search for new cancer-associated genes. *Nature*, 499(7457):214, 2013.

[M+08] Vito MR Muggeo et al. Segmented: an r package to fit regression models with broken-line relationships. *R news*, 8(1):20–25, 2008.

[MA16] Azam Moosavi and Ali Motevalizadeh Ardekani. Role of epigenetics in biology and human diseases. *Iranian biomedical journal*, 20(5):246, 2016.

[MB10] Nicolai Meinshausen and Peter Bühlmann. Stability selection. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(4):417–473, 2010.

[MKC17] Weiguang Mao, Dennis Kostka, and Maria Chikina. Modeling enhancer-promoter interactions with attention-based neural networks. *bioRxiv*, page 219667, 2017.

[MLCPB12] Robert C McLeay, Tom Lesluyes, Gabriel Cuellar Partida, and Timothy L Bailey. Genome-wide in silico prediction of gene expression. *Bioinformatics*, 28(21):2789–2796, 2012.

[MN89] Peter McCullagh and John A Nelder. *Generalized linear models*, volume 37. CRC press, 1989.

[MP43] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[MP11] Flavio Mignone and Graziano Pesole. mrna untranslated regions (utrs). *eLS*, 2011.

[MPJ91] Charlotte H Mason and William D Perreault Jr. Collinearity, power, and interpretation of multiple regression analysis. *Journal of marketing research*, 28(3):268–280, 1991.

[MR71] Friedrich Miescher-Rüsch. *Ueber die chemische Zusammensetzung der Eiterzellen.* 1871.

[MS15] Allison C Mallory and Alena Shkumatava. Lncrnas in vertebrates: advances and challenges. *Biochimie*, 117:3–14, 2015.

[Mug03] Vito MR Muggeo. Estimating regression models with unknown break-points. *Statistics in medicine*, 22(19):3055–3071, 2003.

[MvdGD+13] Graham McVicker, Bryce van de Geijn, Jacob F Degner, Carolyn E Cain, Nicholas E Banovich, Anil Raj, Noah Lewellen, Marsha Myrthil, Yoav Gilad, and Jonathan K Pritchard. Identification of genetic variants that affect histone modifications in human cells. *Science*, 342(6159):747–749, 2013.

[MWR+08] Fabio Mohn, Michael Weber, Michael Rebhan, Tim C Roloff, Jens Richter, Michael B Stadler, Miriam Bibel, and Dirk Schübeler. Lineage-specific polycomb targets and de novo dna methylation define restriction and potential of neuronal progenitors. *Molecular cell*, 30(6):755–766, 2008.

[Nog18] Eishi Noguchi. *DNA Replication Controls: Volume 2*, volume 2. MDPI, 2018.

[NP17] Sarvesh Nikumbh and Nico Pfeifer. Genetic sequence-based prediction of long-range chromatin interactions suggests a potential role of short tandem repeat sequences in genome organization. *BMC bioinformatics*, 18(1):218, 2017.

[O+05] World Health Organization et al. Iarc, international agency for research on cancer who, 2005.

[OC07] James P Orengo and Thomas A Cooper. Alternative splicing in disease. *Advances in experimental medicine and biology*, 623:212–223, 2007.

[OZW09] Zhengqing Ouyang, Qing Zhou, and Wing Hung Wong. Chip-seq of transcription factors predicts absolute and differential gene expression in embryonic stem cells. *Proceedings of the National Academy of Sciences*, pages pnas–0904863106, 2009.

[PBD+13] Len A Pennacchio, Wendy Bickmore, Ann Dean, Marcelo A Nobrega, and Gill Bejerano. Enhancers: five essential questions. *Nature Reviews Genetics*, 14(4):288, 2013.

[PCSS+13] Jennifer E Phillips-Cremins, Michael EG Sauria, Amartya Sanyal, Tatiana I Gerasimova, Bryan R Lajoie, Joshua SK Bell, Chin-Tong Ong, Tracy A Hookway, Changying Guo, Yuhua Sun, et al. Architectural protein subclasses shape 3d organization of genomes during lineage commitment. *Cell*, 153(6):1281–1295, 2013.

[PG98] Roberto Pastor and Eliseo Guallar. Use of two-segmented logistic regression to estimate change-points in epidemiologic studies. *American journal of epidemiology*, 148(7):631–642, 1998.

[PM15] Julia R Pon and Marco A Marra. Driver and passenger mutations in cancer. *Annual Review of Pathology: Mechanisms of Disease*, 10:25–50, 2015.

[PMP04] Giulio Pavesi, Giancarlo Mauri, and Graziano Pesole. In silico representation and discovery of transcription factor binding sites. *Briefings in bioinformatics*, 5(3):217–236, 2004.

[Pra08] Leslie Pray. Dna replication and causes of mutation. *Nature education*, 1(1):214, 2008.

[Pre98] Lutz Prechelt. Early stopping-but when? In *Neural Networks: Tricks of the trade*, pages 55–69. Springer, 1998.

[PVG+11] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

[PY] Alexander Pauls and Josiah A Yoder. Determining optimum drop-out rate for neural networks.

[QB16] Timo Quante and Adrian Bird. Do short, frequent dna sequence motifs mould the epigenome? *Nature Reviews Molecular Cell Biology*, 17(4):257, 2016.

[QX16] Daniel Quang and Xiaohui Xie. Danq: a hybrid convolutional and recurrent deep neural network for quantifying the function of dna sequences. *Nucleic acids research*, 44(11):e107–e107, 2016.

[R+01] Irina Rish et al. An empirical study of the naive bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence*, volume 3, pages 41–46. IBM New York, 2001.

[RDS04] Manuel S Rodriguez, Catherine Dargemont, and Françoise Stutz. Nuclear export of rna. *Biology of the Cell*, 96(8):639–655, 2004.

[RH05] Gajendra PS Raghava and Joon H Han. Correlation and prediction of gene expression level from amino acid and dipeptide composition of its protein. *BMC bioinformatics*, 6(1):59, 2005.

[RHW85] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[RHW86] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533, 1986.

[RIGP18] Alvaro Rada-Iglesias, Frank G Grosveld, and Argyris Papantonis. Forces driving the three-dimensional folding of eukaryotic genomes. *Molecular systems biology*, 14(6):e8214, 2018.

[RKC+13] Debashish Ray, Hilal Kazan, Kate B Cook, Matthew T Weirauch, Hamed S Najafabadi, Xiao Li, Serge Gueroussov, Mihai Albu, Hong Zheng, Ally Yang, et al. A compendium of rna-binding motifs for decoding gene regulation. *Nature*, 499(7457):172, 2013.

[RN16] Stuart J Russell and Peter Norvig. *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited,, 2016.

[Ros58] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[Ros08] AB Rose. Intron-mediated regulation of gene expression. In *Nuclear pre-mRNA processing in plants*, pages 277–290. Springer, 2008.

[RRCB10] William S Rea, Marco Reale, Carmela Cappelli, and Jennifer A Brown. Identification of changes in mean with regression trees: an application to market research. *Econometric Reviews*, 29(5-6):754–777, 2010.

[RSC+15] Sushmita Roy, Alireza Fotuhi Siahpirani, Deborah Chasman, Sara Knaack, Ferhat Ay, Ron Stewart, Michael Wilson, and Rupa Sridharan. A predictive modeling approach for cell line-specific long-range regulatory interactions. *Nucleic acids research*, 43(18):8694–8712, 2015.

[RTL16] Payam Refaeilzadeh, Lei Tang, and Huan Liu. Cross-validation. *Encyclopedia of database systems*, pages 1–7, 2016.

[Rud16] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[SB08] Miho M Suzuki and Adrian Bird. Dna methylation landscapes: provocative insights from epigenomics. *Nature Reviews Genetics*, 9(6):465, 2008.

[SCFK14] Anna Stroynowska-Czerwinska, Agnieszka Fiszer, and Wlodzimierz J Krzyzosiak. The panorama of mirna-mediated mechanisms in mammalian cells. *Cellular and Molecular Life Sciences*, 71(12):2253–2270, 2014.

[SCK04] Meena Kishore Sakharkar, Vincent TK Chow, and Pandjassarame Kangueane. Distributions of exons and introns in the human genome. *In silico biology*, 4(4):387–393, 2004.

[SGG+16] Florian Schmidt, Nina Gasparoni, Gilles Gasparoni, Kathrin Gianmoena, Cristina Cadenas, Julia K Polansky, Peter Ebert, Karl Nordstroem, Matthias Barann, Anupam Sinha, et al. Combining transcription factor binding affinities with open-chromatin data for accurate gene expression prediction. *Nucleic acids research*, 45(1):54–66, 2016.

[SGK17] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. *arXiv preprint arXiv:1704.02685*, 2017.

[SH10] Carla Sawan and Zdenko Herceg. Histone modifications and cancer. In *Advances in genetics*, volume 70, pages 57–85. Elsevier, 2010.

[Sha13] Greg Shaw. Polymorphism and single nucleotide polymorphisms (snp s). *BJU international*, 112(5):664–665, 2013.

[SHK+14] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[SKK+03] Toshiyuki Shiraki, Shinji Kondo, Shintaro Katayama, Kazunori Waki, Takeya Kasukawa, Hideya Kawaji, Rimantas Kodzius, Akira Watahiki, Mari Nakamura, Takahiro Arakawa, et al. Cap analysis gene expression for high-throughput analysis of transcriptional starting point and identification of promoter usage. *Proceedings of the National Academy of Sciences*, 100(26):15776–15781, 2003.

[SL87] Paul M Sharp and Wen-Hsiung Li. The codon adaptation index-a measure of directional synonymous codon usage bias, and its potential applications. *Nucleic acids research*, 15(3):1281–1295, 1987.

[SLA12] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pages 2951–2959, 2012.

[SLJD12] Amartya Sanyal, Bryan R Lajoie, Gaurav Jain, and Job Dekker. The long-range interaction landscape of gene promoters. *Nature*, 489(7414):109, 2012.

[SLT+04] Jonathan Sebat, B Lakshmi, Jennifer Troge, Joan Alexander, Janet Young, Pär Lundin, Susanne Månér, Hillary Massa, Megan Walker, Maoyen Chi, et al. Large-scale copy number polymorphism in the human genome. *Science*, 305(5683):525–528, 2004.

[SM14] Rajen Dinesh Shah and Nicolai Meinshausen. Random intersection trees. *The Journal of Machine Learning Research*, 15(1):629–654, 2014.

[SP97] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.

[SS90] Thomas D Schneider and R Michael Stephens. Sequence logos: a new way to display consensus sequences. *Nucleic acids research*, 18(20):6097–6100, 1990.

[SS09] Ashok N Srivastava and Mehran Sahami. *Text mining: Classification, clustering, and applications*. Chapman and Hall/CRC, 2009.

[SS18] Florian Schmidt and Marcel H Schulz. On the problem of confounders in modeling gene expression. *Bioinformatics*, page bty674, 2018.

[Sto00] Gary D Stormo. Dna binding sites: representation and discovery. *Bioinformatics*, 16(1):16–23, 2000.

[SW89] GAF Seber and CJ Wild. Nonlinear regression: Wiley series in probability and mathematical statistics, 1989.

[SW15] James H Stock and Mark W Watson. *Introduction to econometrics*. 2015.

[SYPM16] Shashank Singh, Yang Yang, Barnabas Poczos, and Jian Ma. Predicting enhancer-promoter interaction from genomic sequence with deep neural networks. *bioRxiv*, page 085241, 2016.

[Tib96] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

[TOHTS13] Aika Terada, Mariko Okada-Hatakeyama, Koji Tsuda, and Jun Sese. Statistical significance of combinatorial regulations. *Proceedings of the National Academy of Sciences*, 110(32):12996–13001, 2013.

[TZ81] Asher Tishler and Isreal Zang. A maximum likelihood method for piecewise regression models with a continuous dependent variable. *Applied Statistics*, pages 116–124, 1981.

[VCL+17] Jimmy Vandel, Oceane Cassan, Sophie Lebre, Charles-Henri Lecellier, and Laurent Brehelin. Modeling transcription factor combinatorics in promoters and enhancers. *bioRxiv*, page 197418, 2017.

[Voj16] Ján Vojt. Deep neural networks and their implementation. 2016.

[WC+53] James D Watson, Francis HC Crick, et al. Molecular structure of nucleic acids. *Nature*, 171(4356):737–738, 1953.

[WCW14] John W Whitaker, Zhao Chen, and Wei Wang. Predicting the human epigenome from dna motifs. *Nature methods*, 12(3):265, 2014.

[Wei07] RA Weinberg. Maintenance of genomic integrity and the development of cancer. *The Biology of Cancer. New York: Garland Science*, pages 463–526, 2007.

[WS04] Wyeth W Wasserman and Albin Sandelin. Applied bioinformatics for the identification of regulatory elements. *Nature Reviews Genetics*, 5(4):276, 2004.

[WSW53] Maurice Hugh Frederick Wilkins, Alex R Stokes, and Herbert R Wilson. Molecular structure of nucleic acids: molecular structure of deoxypentose nucleic acids. *Nature*, 171(4356):738, 1953.

[Wu80] Carl Wu. The 5âĂš ends of drosophila heat shock genes in chromatin are hypersensitive to dnase i. *Nature*, 286(5776):854, 1980.

[WZW+12] Ya-Mei Wang, Ping Zhou, Li-Yong Wang, Zhen-Hua Li, Yao-Nan Zhang, and Yu-Xiang Zhang. Correlation between dnase i hypersensitive site distribution and gene expression in hela s3 cells. *PloS one*, 7(8):e42414, 2012.

[XYF+07] Hualin Xi, Yong Yu, Yutao Fu, Jonathan Foley, Anason Halees, and Zhiping Weng. Analysis of overrepresented motifs in human core promoters reveals dual regulatory roles of yy1. *Genome research*, 17(6):798–806, 2007.

[YCBL14] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328, 2014.

[YGSL07] Yuan Yuan, Lei Guo, Lei Shen, and Jun S Liu. Predicting gene expression from sequence: a reexamination. *PLoS computational biology*, 3(11):e243, 2007.

[YLM+06] Xueping Yu, Jimmy Lin, Tomohiro Masuda, Noriko Esumi, Donald J Zack, and Jiang Qian. Genome-wide prediction and characterization of interactions between transcription factors in saccharomyces cerevisiae. *Nucleic acids research*, 34(3):917–927, 2006.

[ZELG16] Haoyang Zeng, Matthew D Edwards, Ge Liu, and David K Gifford. Convolutional neural network architectures for predicting dna–protein binding. *Bioinformatics*, 32(12):i121–i127, 2016.

[ZH05] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005.

[Zho12] Zhi-Hua Zhou. *Ensemble methods: foundations and algorithms*. Chapman and Hall/CRC, 2012.

[ZSRU91] PA Zimmerman, ML Spell, J Rawls, and TR Unnasch. Transformation of dna sequence data into geometric symbols. *BioTechniques*, 11(1):50–52, 1991.

[ZT15] Jian Zhou and Olga G Troyanskaya. Predicting effects of noncoding variants with deep learning–based sequence model. *Nature methods*, 12(10):931, 2015.

[ZTY+18] Jian Zhou, Chandra L Theesfeld, Kevin Yao, Kathleen M Chen, Aaron K Wong, and Olga G Troyanskaya. Deep learning sequence-based ab initio prediction of variant effects on expression and disease risk. *Nature genetics*, 50(8):1171, 2018.