PATTERNS AND ALGORITHMS IN HIGH-THROUGHPUT SEQUENCING COUNT DATA

Alessandro Mammana

Dissertation zur Erlangung des Grades
eines Doktors der Naturwissenschaften (Dr. rer. nat.)
am Fachbereich Mathematik und Informatik
der Freien Universität Berlin

September 2015

Gutachter: Prof. Dr. Martin Vingron
Zweitgutachter: Prof. Dr. Dr. Thomas Lengauer

Tag der Disputation: 19.2.2016

Preface

Collaborations and publications

Parts of this thesis grew out of collaborations with other researchers and can also be found in scientific publications, although in a different format. The project presented in Chapter 3 carries significant contributions from Martin Vingron and Ho-Ryun Chung and it was published in the journal Bioinformatics [1]. The idea of detecting nucleosomes from histone modification ChIP-seq data, in fact, was originally proposed by Martin and the clustering analysis of Section 3.3.7 is largely due to Ho. The work presented in Chapter 4 was published in the journal Genome Biology [2] with the help of Ho-Ryun Chung. Chapter 5 originates from a collaboration with Sebastiaan Meijsing, who provided the ChIP-exo data for the glucocorticoid receptor and laid the basis for this project. Many ideas about the computational method, moreover, especially at the beginning, were contributed by Ho-Ryun Chung. The bamsignals package, which underlies the work presented in Chapters 4 and 5 and which was published on Bioconductor [3], was implemented in collaboration with of Johannes Helmuth, who helped me to extend the software to paired-end sequencing libraries. Finally, this thesis benefited considerably from the proofreading of Anna Ramisch, Edgar Steiger, Ho-Ryun Chung, Juliane Perner, Matt Huska, Philipp Benner, Robert Schöpflin, Sebastiaan Meijsing and Wolfgang Kopp, to whom I am truly grateful.

Acknowledgements

Throughout my PhD I received substantial help from many people. First and foremost, from my supervisors Ho-Ryun Chung and Martin Vingron. I thank Ho for the independence and trust that he gave me, for encouraging me to pursue my ideas, and for the many stimulating conversations that we had together. I still would not understand epigenomics without his help, and my knowledge about data analysis would be much more modest. I thank Martin for his supervision, for his friendly advice, and for making many things possible.

I thank all current and past members of Vingron's and Chung's group for the lively and collaborative environment in the institute. I am grateful to Mike Love for his invaluable help with statistics and data analysis, for the many stimulating discussions about bioinformatics, and for convincing me to use the R programming

language, thus defeating my stubborn repulsion. I am grateful to Matt Huska for his help with R, for correcting terrible manuscripts, and for the many interesting conversations. I am grateful to Johannes Helmuth for his help in the implementation of the fastest pile-up-BAM in the south. I thank all members of the DREAM8-team and the legendary anti-Peter Mathekalender squad, including Peter Arndt, for the great learning experience that we had together. I am grateful to Alena van Bömmel, Anna Ramisch, Brian Caffrey, Edgar Steiger, Johannes Helmuth, Juliane Perner, Mahsa Ghanbari, Matt Huska, Mike Love, Mohammad Sadeh, Robert Schöpflin, Wolfgang Kopp and Xinyi Yang for putting up with my bad jokes and for making my time in Berlin so enjoyable.

Lastly, a special thank goes to my parents, Anna and Giacomo, for their support, their understanding, and their unlimited love, which have been fundamental in these years.

Contents

Chapter 1

Introduction

In an interview for a bookstore company in 1993 Donald E. Knuth, one of the fathers of computer science, said about bioinformatics:

> "There's millions and millions of unsolved problems. Biology is so digital, and incredibly complicated, but incredibly useful. [...] Biology easily has 500 years of exciting problems to work on." [4]

So far his predictions seem to be accurate. In particular, the availability of genomic sequences presented scientists with a large number of digital problems, which fostered considerable advances in formal sciences, such as computer science, machine learning and statistics, as well as in life sciences. Many of these problems could be formalized as the discovery, detection or statistical analysis of patterns in a long sequence. This is the genetic code: in humans, a signal of about three billion symbols in an alphabet of four letters. However today many of these problems, such as read mapping, sequence alignment, motif discovery, or alignment statistics, are already very well studied.

Opportunely, new experimental techniques, such as ChIP-seq, are providing a new type of digital sequences. These are count signals: sequences as long as the genetic code, but with the natural numbers as an alphabet. Count signals form the basis for a new kind of challenges with at least three levels of complexity. At the biological level, ChIP-seq and count signals shed light on gene regulation: an intricate network of molecular processes that orchestrates the genetic code and determines the identity of a cell. These processes can now be systematically assayed and analyzed computationally. At the data analysis level, the code hidden in the count signals is extremely complex due to experimental noise and the heterogeneity of cell mixtures. This calls for the development of novel statistical approaches to explore, characterize and confidently detect the patterns of interest. Lastly, at the computational level, an unprecedented amount of data is being produced and needs to be analyzed efficiently. Unlike the genetic code, in fact, profiling the regulatory mechanisms in a cell type often results in more than one count signal, and these signals also differ from one cell type to the next.

The purpose of this thesis is to formalize and address pattern discovery and pattern detection problems in count signals. These problems are biologically relevant

in the context of gene regulation, for understanding how genes are regulated and what characterizes a cell type. The proposed solutions and the main contributions of this work consist in the design and implementation of algorithms for the analysis of these biological datasets.

This chapter serves as a general introduction to the thesis. In particular, Section 1.1 gives a brief overview about gene regulation and Section 1.2 introduces the ChIP-seq technique and count signals; these are the common biological motivation and the common type of input data of the proposed algorithms. Finally, Section 1.3 describes how the thesis is structured.

1.1 Biological context

1.1.1 The genome

The genome has been called the book of life, because it is what determines almost every feature of a living organism. It is also the micro-universe where all the biological phenomena studied in this work take place. To describe where events occur in this universe it is necessary to clarify its topology.

The eukaryotic genome resides in the cell nucleus, packaged into units called chromosomes and constituted by double-stranded DNA molecules. A DNA strand is a long chain of nucleotides which can be seen as a text written with only four symbols: adenine (A), cytosine (C), guanine (G) and thymine (T). Owing to the asymmetry of the chemical bond between two adjacent nucleotides, in a DNA strand it is always possible to distinguish a start and an end, called respectively the 5' end and the 3' end. The double-stranded DNA contained in the chromosomes is composed of two complementary strands. Complementary means that they have the same number n of nucleotides, that the i-th nucleotide in one strand is bound to the $(n-i+1)$-th nucleotide in the other strand, and that this bound obeys the base pairing rules (adenine is paired with thymine and cytosine is paired with guanine). For more information about the structure of double-stranded DNA see Watson & Crick [5].

Today the human genome sequence, as well as that of many other species, is almost entirely known [6, 7] and reference genome assemblies are available. The assembly specifies the nucleotide sequence of each chromosome and defines a forward and a reverse strand. A genomic coordinate, therefore, consists of a chromosome, a strand, and the number of base pairs from the start of the forward strand.

1.1.2 Transcription

Transcription is the first step in the process that propagates the information in the genome outside the cell nucleus and is one of the most important steps in gene regulation. The central dogma of molecular biology [8], in fact, states that every protein originates from a corresponding region in the genome called gene. During transcription a protein complex called RNA polymerase reads the gene in the 5'-to-

3' direction and creates an RNA molecule called pre-mRNA (see Figure 1.1). This molecule acts as a sort of temporary storage device for transferring information from the cell nucleus to the cytoplasm. Inside the cell nucleus the pre-mRNA is transformed into an mRNA by splicing and polyadenylation. In the cytoplasm the mRNA undergoes further steps (translation) that eventually lead to the synthesis of a protein.

Transcription is a three-step process, consisting of initiation, elongation and termination. The initiation of transcription is mediated by a protein complex called the preinitiation complex, and the exact site where the RNA polymerase starts reading the gene is called transcription start site (TSS). Next, the RNA polymerase elongates along the gene body and finally dissociates from the DNA.

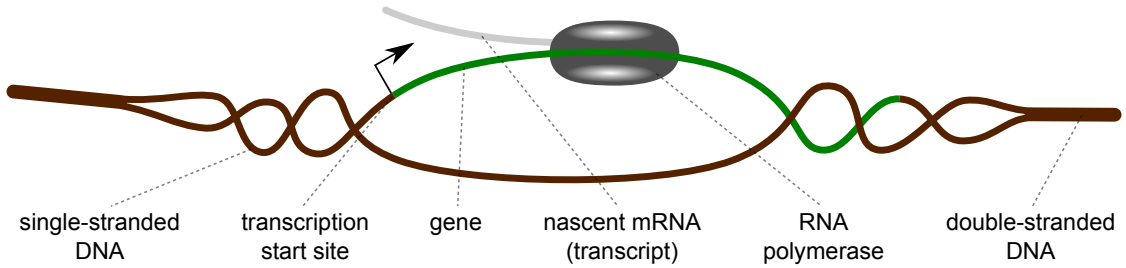

Figure 1.1: *Gene transcription. The drawing shows the RNA polymerase elongating through the body of a gene. The gene being transcribed is represented as the green segment in the single-stranded DNA region. The transcription start site is represented as an arrow. The transcript, shown in light-gray, is almost a copy of the gene and it is free to leave the cell nucleus and to be translated into a protein.*

1.1.3 Gene regulation

If the genome is the book of life, in a multicellular organism each cell has almost the same book, but reads different pages. In fact a human is not a homogeneous mass of cells, but rather a highly structured organism composed of distinct cell types, such as white blood cells, neurons or muscle cells. What makes these cells specialized is, to a large extent, their transcriptional program. In fact, even if all cells share an almost identical genome, their genes are transcribed at different rates. This has a direct impact on gene expression, i.e. how many proteins from each gene are present in the cell, which in turn explains a wide range of phenotypic differences. There are also other mechanisms that regulate gene expression and that occur after transcription, such as microRNA- and lncRNA- mediated regulation, mRNA degradation and post-translational modifications [9–12]. Gene regulation is the field of molecular biology that studies these yet poorly understood mechanisms. The central questions of this field are how are genes regulated and what determines the identity of a cell. The computational approaches presented in this thesis are an effort towards the answer to these questions.

1.1.4 Transcription factors

Transcription factors (TFs) are proteins of fundamental importance in gene regulation. The distinctive feature of these proteins is that they can bind to the genome and subsequently activate or repress the transcription of one or more target genes. Transcription factors do not bind everywhere in the genome, but they are attracted to specific genomic locations, called binding sites, where certain DNA sequences occur. Each TF recognizes a set of similar sequences, collectively called a motif, which can be 4 to 15 base pairs long. However, motifs are not the only determinants of TF binding. The accessibility of the DNA and the presence of other proteins can play a major role.

TFs bind to promoters or enhancers. Promoters are located right upstream the transcription start site, while enhancers can be even millions of base pairs away and brought in proximity of the transcription start site by DNA looping [13]. After binding, there are many mechanisms by which a TF can affect transcription. TFs that act as activators, for instance, can directly or indirectly recruit (attract) members of the preinitiation complex, and thereby facilitate the initiation of transcription. In contrast, TFs acting as repressors can block an activator by steric hindrance, i.e., they compete with the activator for the same binding site and prevent it from binding. Very often TFs do not act in isolation, but they need to interact with other molecules, called cofactors, in order to perform their regulatory role.

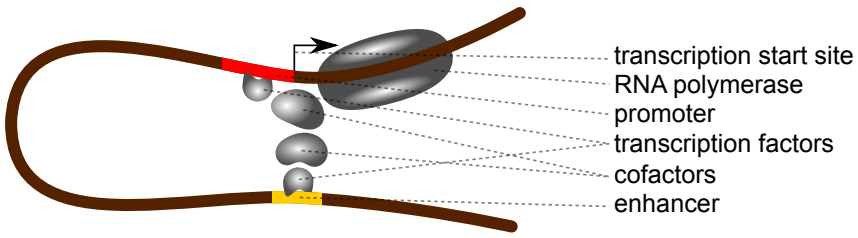

Figure 1.2: *TFs regulate genes by binding to promoters or enhancers. Promoters are located right upstream of the TSS of the regulated gene, while enhancers can be thousands of bases away from the TSS. The regulatory effect is achieved by interacting with RNA polymerase and other proteins responsible for initiating transcription, often through intermediate proteins called cofactors.*

The detection of TF binding sites is a first important step in understanding how genes are differentially regulated in different cell types. Some computational methods base their predictions on the presence of known sequence motifs [14, 15]. These approaches, however, have several limitations. First, they require prior biological knowledge about the TF of interest, second, they cannot explain the differences among cell types and, most importantly, they do not take into account the accessibility of the DNA and the combinatorial effect of other cofactors. Other computational methods [16, 17] are based on the ChIP-seq protocol (see Section 1.2), which provides a more direct measurement of protein binding. However these approaches

suffer from limited resolution. In Chapter 5 we present a novel approach that unifies both data sources and provides accurate predictions of TF binding sites.

1.1.5 The epigenome

Without altering the DNA sequence, cells can still modify certain molecules in close proximity with the DNA, which, in turn, are related to gene expression [18]. Collectively, these modifications constitute the epigenome, and they can be seen as a mechanism for deriving different interpretations from the same genetic code. While the genome is passed on to daughter cells with high fidelity, the epigenome is not as stable. How and to which extent certain epigenetic modifications can be inherited across cell divisions is still a matter of debate [19]. Histones and histone modifications, presented in Subsection 1.1.6, are an important component of the epigenome. There are also other modifications that constitute the epigenome, notably DNA methylation [20]. However, in this work the word epigenome will almost always refer to histones and histone modifications.

1.1.6 Histones

Histones are proteins found inside the cell nucleus of eukaryotic organisms. These proteins are extremely well conserved throughout evolution, which suggests that they are important for the cell [21]. One of their main roles is to organize and compact the genetic material inside the nucleus. In other words, they are responsible for fitting and organizing an entire human genome, which is about 2 meters long, into a cellular compartment with the diameter of about 6 micrometers. This is achieved through organizational units called nucleosomes: a complex formed by a stretch of DNA of about 147 base pairs and by two copies each of the four core histones H2A, H2B, H3 and H4 [22] (see Figure 1.3). The histone octamer serves as a sort of spool around which the genome can wind, forming a structure resembling beads on a string. Moreover, this structure can wind around itself multiple times and achieve higher degrees of compaction.

The term chromatin refers to the structure formed by histones and DNA (as well as other components), and it is traditionally divided into euchromatin and heterochromatin. Euchromatin is characterized by a low degree of compaction, it tends to localize at the center of the cell nucleus, and the genes that it hosts can be easily transcribed. Heterochromatin, by contrast, is highly condensed chromatin located at the periphery of the nucleus. Heterochromatic regions tend to harbor fewer genes, and those genes tend to be repressed. In general, nucleosomes tend to inhibit gene transcription by making the DNA less accessible [23].

What determines the position of nucleosomes along the genome is still unclear and is the focus of ongoing research. On one hand it seems that nucleosome formation is driven, at least partially, by the DNA sequence [24]. On the other hand many molecular processes, such as transcription initiation and elongation, and many enzymes, such as ATP-dependent chromatin modifiers, are known to be responsible

for nucleosome formation and eviction [25–27]. In Chapter 3 we present a method that uses ChIP-seq data (see Section 1.2) to infer where nucleosomes are bound.

The role of histones in gene regulation is much more complex than just a general repressive mechanism. It is not only important where a nucleosome forms, but also the presence or absence of certain chemical groups in specific residues of the histones, called histone modifications (or marks). Modified residues tend to occur in domains of the histones which protrude from the nucleosome and are therefore called histone tails (see Figure 1.3). Histone modifications are named after the histone where they occur, the position of the modified residue within the histone, and the chemical group that is covalently bound to the residue. For instance, H3K4me3 denotes the addition of three methyl groups in the fourth residue of histone H3, which is a lysine.

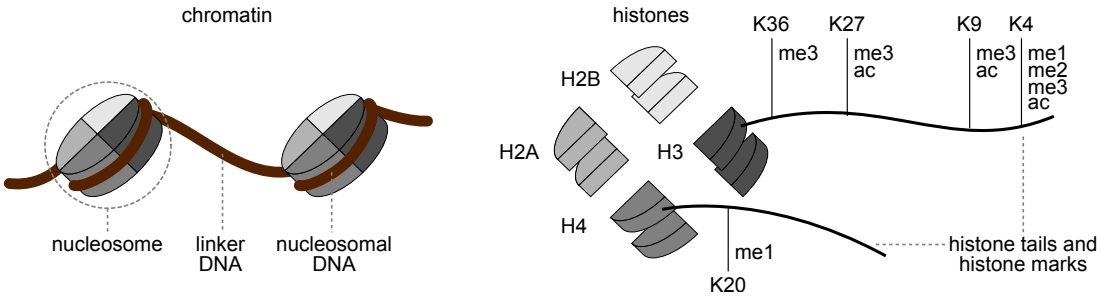

Figure 1.3: *Nucleosomes, histones and histone marks. A nucleosome is a complex formed by histone proteins (in gray) and DNA (in brown) and is the fundamental repeating unit of the chromatin. Each nucleosome is formed by two copies of four types of histones: H2A, H2B, H3 and H4. Certain residues of the histones, especially in their tail domains, can be covalently modified and carry specific post-translational modifications. The most common marks have been represented by specifying the residue where they occur (K stands for lysine, and the number that follows specifies how far the residue is form the N-terminus of the protein), and the covalent modifications that are typically studied (me1, me2, and me3 stand for mono-, di- and tri-methylation and ac stands for acetylation). Note that not only the tail domains of the histones can be modified and not only lysine residues. Note also that all histones have tails. Here, for display purposes, only two tails have been drawn.*

Because histone modifications can be recognized, added and removed by other proteins, it has been proposed that they constitute a sort of code which orchestrates the regulatory mechanisms in the chromatin. This idea is often referred to as the histone code hypothesis [28]. Today many histone marks have been related to regulatory mechanisms and to gene transcription [29, 30]. The modifications H3K4me3 and H3K36me3, for instance, are strongly associated to transcription initiation [31, 32] and elongation [33, 34], respectively, and are therefore considered active marks. Also acetylations are usually active marks, because they make the nucleosome less stable and therefore the DNA more accessible [35, 36]. The modifications H3K27me3 and H3K9me3, on the other hand, are considered repressive

histone mark	relation to transcription	typical location
H3K4me3	activation	promoters
H3K4me1	activation	promoters,enhancers
H3K36me3	activation	gene bodies
H3K27ac	activation	promoters,enhancers
H3K27me3	repression	repressed genes
H3K9me3	repression	heterochromatin

Table 1.1: *Some histone modifications and their association to transcription.*

marks because they are involved in mechanisms that repress gene transcription [31, 37, 38]. Table 1.1 summarizes some of the most important and best understood histone modifications together with their association to transcription.

A concept related to the histone code is the concept of chromatin state: a set of histone marks that tend to occur in the same genomic region [39]. Chromatin states are a convenient abstraction mainly for two reasons. First, they provide a grammar for studying the role played by combinations of histone marks, rather than by histone marks in isolation. Second, they allow a compact description of the highly complex histone code. In fact, considering every possible combination of histone marks along with their intensities can be impractical. In many applications it is possible to reduce all possible configurations of histone marks to a small number of states, which can be easily interpreted and analyzed. In Chapter 4 we present an algorithm that automatically detects biologically relevant chromatin states and segments the genome based on which marks are present in each region.

1.2 Measurement of protein binding

1.2.1 ChIP-seq

Chromatin Immunoprecipitation followed by Sequencing (ChIP-seq) is an experimental method that measures from a given sample of cells the abundance of a specific protein (a transcription factor, a histone or a histone with a specific mark) along the entire genome [40]. Omitting many technical details, the experiment works as follows:

1. A large number of cells is fixed using formaldehyde. This process, called crosslinking, stabilizes the bonds between proteins and DNA.

2. The chromatin of the fixed cells is broken into small fragments through sonication. Sonication consists in applying sound energy to agitate the molecules in the sample.

3. Specific antibodies that bind only to the protein of interest are added to the sample, and the complexes formed by the antibody, the protein of interest

and the DNA fragment are isolated from the rest. This process is called immunoprecipitation.

4. The proteins are removed from the sample and purified DNA fragments are prepared for the sequencing phase. This process is called library preparation. An important part of library preparation is DNA amplification, which consists in duplicating the available DNA multiple times.

5. A sequencing machine decodes the stretches of nucleotides at the 5' end of the DNA fragments, which are called reads.

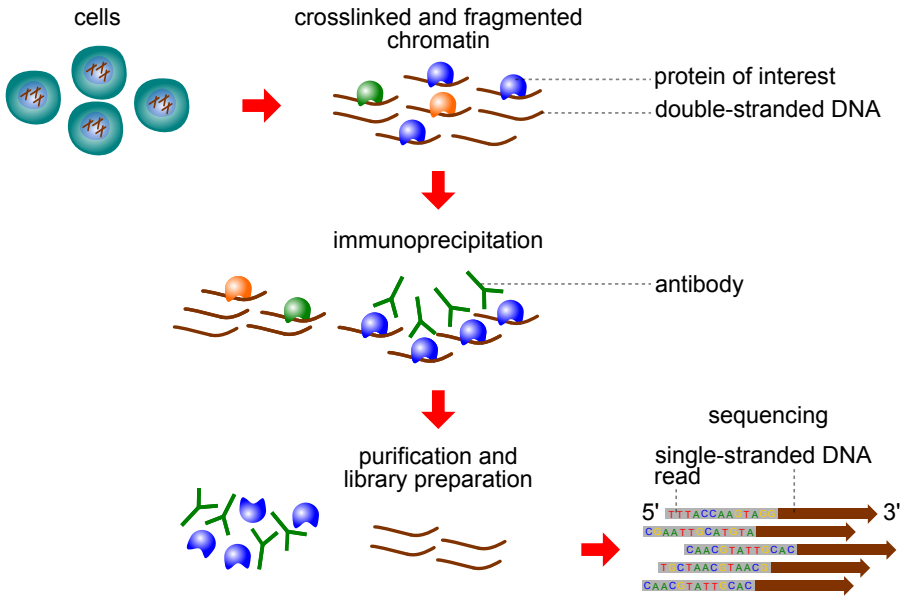

Figure 1.4: *The ChIP-seq protocol. First, the chromatin is extracted from the nucleus of a large number of cells. Second, the chromatin is fragmented, which results in a large number of double-stranded DNA fragments. Some of these fragments will be bound to the protein of interest and others not. Next, specific antibodies bind to the protein of interest and can be subsequently isolated. This separates the antibody-protein-DNA complexes from the rest. Next, the DNA is purified and prepared for sequencing. Finally, the 5' extremities of single-stranded DNA fragments (represented as brown arrows showing the 5'-to-3' direction) are decoded by a sequencing machine. These short sequences (typically from 20 to 50 base pairs long) are called reads.*

There exist many variations of the ChIP-seq protocol. Chapter 5 presents a variant called ChIP-exo [41]. For the purposes of this introduction, the differences between ChIP-exo and ChIP-seq can be ignored.

1.2.2 Read mapping

After sequencing, the reads from a ChIP-seq experiment are mapped, that is, their original location in the genome is determined (see Figure 1.5). Tools called read mappers search the reference genome for a subsequence matching the read. Once a hit is found, the read is assigned a genomic coordinate consisting of a chromosome identifier, the position from the start of the chromosome, and the strand. Typically, most of the reads contain enough information for a unique position in the genome to be identified. When the reads are too short compared to the reference genome, when they contain too many sequencing errors or when they come from repetitive regions of the genome it might be impossible to map the read confidently.

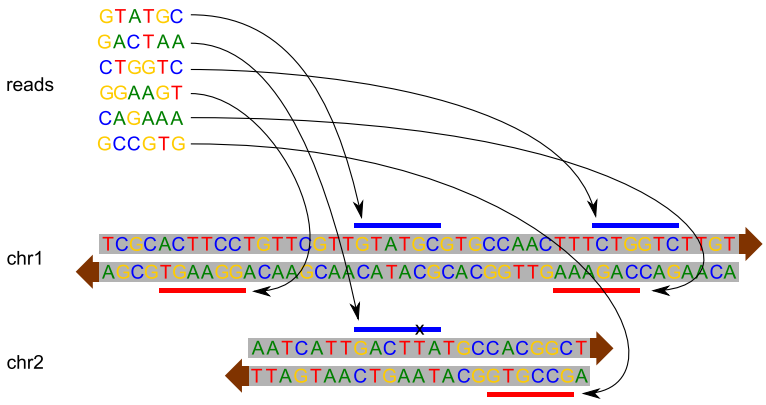

Figure 1.5: *Read mapping. In this toy example, the reads at the top are mapped to the reference genome at the bottom consisting of two chromosomes. This is done by searching for the subsequence of the reference genome that matches the read. Such a match can occur in the forward or in the reverse strand of a chromosome, which is drawn respectively with blue and red rectangles. A match does not necessarily imply that all bases are identical; there can be small differences, such as mismatches (drawn with an x). These can be due to sequencing errors or differences between the reference genome and the genome of the cells where the reads originate from.*

1.2.3 Count signals

The algorithms presented in this work process count signals: sequences of natural numbers obtained by counting how many reads are mapped at each genomic coordinate. The count signals obtained from ChIP-seq reads, therefore, indicate where and to which extent proteins bind to the genome. Count signals can also be obtained from other important experimental protocols, such as DNAse-seq and RNA-seq [42, 43]. In DNAse-seq, a count signal shows to which extent the DNA is accessible for proteins such as TFs and the preinitiation complex. In RNA-seq read counts measure the abundance of RNA originating from a certain DNA region. Count signals can be conveniently displayed in a genome browser, where the horizontal axis shows

the position in the forward strand of a chromosome, and the vertical axis shows the read count (see Figure 1.6).

There are many possible ways of assigning reads to genomic coordinates. When a read is mapped, in fact, it overlaps more than one base in the reference genome, and a choice has to be made as to which elements of the count signal should be increased by one. Moreover, count signals can have different resolutions. For instance, instead of counting reads for each base pair it is very common to partition the reference genome into bins of a fixed size, thereby aggregating adjacent base pairs. If each bin has a length of 100 base pairs, the resulting count signal is 100 times shorter. This can be convenient for data analysis and visualization. Finally, count signals can be strand-specific. That is, from one ChIP-seq sample two count signals can be defined by considering reads mapping to the forward and reverse strand separately. Each of these signals is a strand-specific signal. The exact way count signals are defined is application-specific and will be clarified in each chapter.

The bamsignals package [3] was developed to obtain count signals from mapped reads efficiently, allowing for all the above-mentioned counting criteria and many others. Even though bamsignals is not presented in this thesis because it does not constitute a conceptual innovation, it is an important building block for the algorithms presented in the next chapters, as well as a technical and practical advancement for the bioinformatics community.

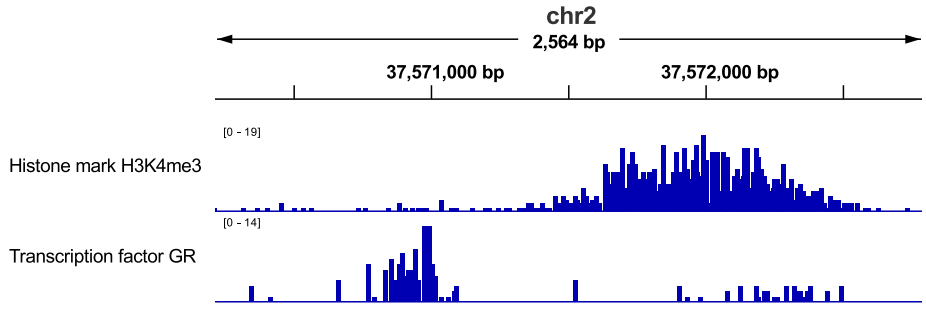

Figure 1.6: *Example of count signals in a genome browser. On top the portion of the reference genome under consideration can be read: in this example a window of 2564 bases in chromosome "chr2", about 37.5 million bases from the start of the chromosome. Below two count signals are shown, often called tracks. In both tracks, the vertical axis shows how many reads from a ChIP-seq experiment are mapped to a certain position. Here, reads mapped to the forward and reverse strand are summed together. The first track shows the abundance of a certain histone modification, the second one shows the abundance of a certain transcription factor.*

1.3 Thesis overview

This chapter explained the biological motivations behind the development of computational approaches for ChIP-seq count data and what this data represents. Chapter

2, instead, provides the mathematical and computational foundations of the proposed methods. The remainder of this thesis presents the main contributions. Each contribution is a computational method that addresses a different biological problem in the context of gene regulation. What unifies them is that the biological entities of interest manifest themselves as a pattern in the count signals derived from ChIP-seq data. Figure 1.7 summarizes the three problems addressed in the following chapters in terms of a biological process and its corresponding pattern.

In Chapter 3 the nucleosome detection problem is presented. This problem consists in identifying where nucleosome are positioned in the genome from one or more ChIP-seq samples. The proposed solution is based on efficient signal processing techniques and statistical methods. The results are used for exploring how histone marks are related to nucleosome positions.

Chapter 4 discusses the chromatin segmentation problem. Here the input data is a set of ChIP-seq experiments for different histone marks and the patterns to be automatically discovered and detected are the chromatin states: recurrent configuration of the chromatin resulting in characteristic combinations of histone mark abundances. The proposed solution is based on the probabilistic modeling of multivariate count data and on unsupervised learning techniques.

In Chapter 5 the transcription factor footprint discovery problem is introduced. The pattern to be characterized and detected is the binding of a transcription factor to the genome, which can be observed from a very specific pattern in the ChIP-exo read abundances and from the presence of a sequence motif in the underlying genomic sequence. Genome sequence and ChIP-exo read counts are, therefore, the input data for this problem, which is approached by combining probabilistic assumptions and classification methods.

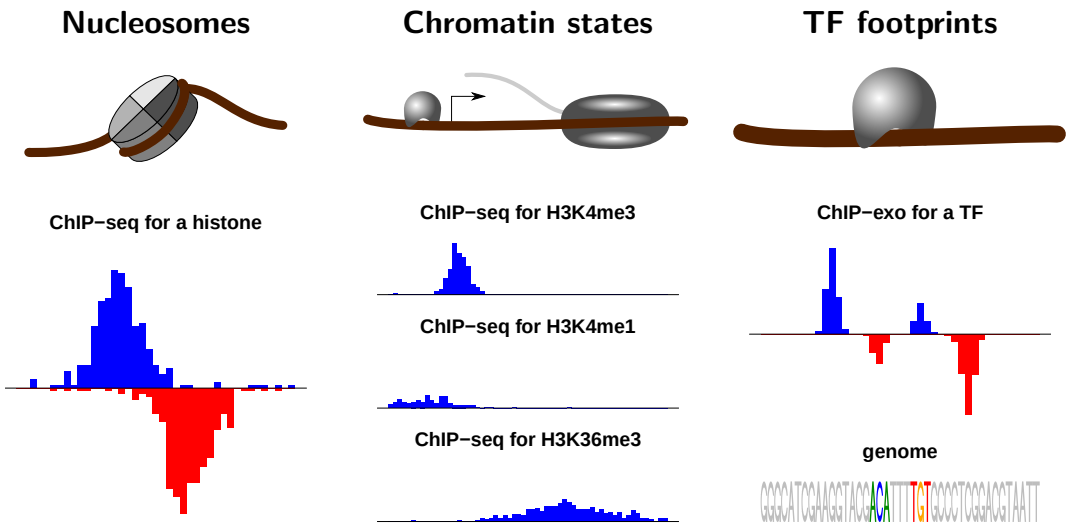

Figure 1.7: *Three biological processes and their patterns in the count signals. On the left, the biological entity of interest is the nucleosome. In a ChIP-seq experiment directed against a histone protein, the presence of a nucleosome results in two related gaussian-like peaks in the strand-specific count signals. On the middle, the goal is to discover patterns in multiple count signals for histone modifications. These patterns can be related to transcription initiation, transcription elongation, enhancers, and other important biological processes. On the right the binding between transcription factors and DNA is studied at a high resolution. Binding sites are characterized by a very information-rich pattern in the count signal obtained from ChIP-exo data, as well as in the underlying genomic sequence.*

Chapter 2

Mathematical prerequisites

2.1 Notation

We outline here certain notational conventions that we adopt throughout the thesis. Upper case symbols can denote random variables or sets and symbols in boldface denote vectors. For example, random vectors are denoted by upper case boldface symbols and scalar numbers or scalar functions by lower case letters in normal type. Matrices are denoted in the same way as vectors, as they can be thought of as vectors of rows. Finally, character constants are denoted in monospace. There are few exceptions to these rules motivated by widely adopted conventions (for instance, in Subsection 5.2.9 the cumulative distribution function of a gaussian random variable is denoted by F).

Most of the formulas in the next chapters illustrate the behaviour of algorithms, therefore, a large number of symbols is used to denote data, variables and parameters. To keep the notation simple, in some cases, the same symbol is used in two chapters with two different meanings (for instance, the variable μ_j in Subsection 3.2.6 is not related to the variable μ_j in Subsection 4.2.3). Moreover, the assignment operator is denoted simply by $=$. The ambiguities are resolved by the adjacent text.

2.2 Probabilistic models

In Figure 1.7 we saw how the biological phenomena under consideration manifest themselves as specific patterns in the data. Probability theory provides a language to describe these patterns and to formalize the analysis of the data as a pattern discovery or pattern detection problem. Moreover, the models that we will consider are interpretable, which means that not only the predictions made by the models are useful, but also the models themselves can be used to reason about the underlying biological phenomena. The topics presented here are by no means an introduction to probability theory (for this purpose see [44], for instance), but rather a mathematical introduction to the contributions of this thesis.

2.2.1 Position probability matrices

In Subsection 1.1.4 we saw that transcription factors target genomic locations where a specific set of DNA sequences occur, and in Chapter 5 we address the problem of characterizing this set. Position probability matrices (PPMs) are among the simplest and most used models to describe the sequence preferences of a transcription factor [45]. Let $\boldsymbol{s} = (s_1, s_2, ...s_n)$ denote a sequence of n symbols where $s_i \in \{\mathtt{a}, \mathtt{c}, \mathtt{g}, \mathtt{t}\}$. The key assumption of this model is that each position of the sequence is characterized by a random variable independent from the others. For each position i the observed symbol follows a categorical distribution with outcomes $\mathtt{a}, \mathtt{c}, \mathtt{g}, \mathtt{t}$ and probabilities $p_{i\mathtt{a}}, p_{i\mathtt{c}}, p_{i\mathtt{g}}, p_{i\mathtt{t}}$, where all the coefficients $p_{i\alpha}$ are positive and their sum equals 1.

Let $\boldsymbol{S}$ be the random vector representing the whole sequence and S_i the random variable corresponding to the i-th categorical distribution. Because of the independence assumption, the probability of the observed sequence $\boldsymbol{s}$ can be written as:

$$\mathrm{P}\{\boldsymbol{S} = \boldsymbol{s}\} = \prod_{i=1}^{n} \mathrm{P}\{S_i = s_i\} = \prod_{i=1}^{n} \sum_{\alpha \in \{\mathtt{a},\mathtt{c},\mathtt{g},\mathtt{t}\}} p_{i\alpha} \cdot [s_i = \alpha],$$

where the square brackets evaluate to 1 if the enclosed expression is true and to 0 otherwise (Iverson brackets). The coefficients $p_{i\alpha}$, which characterize the random variable S_i, can be arranged as a matrix $\boldsymbol{p} = (\boldsymbol{p}_1, \boldsymbol{p}_2, ..., \boldsymbol{p}_n)$ of n rows and 4 columns $\boldsymbol{p}_i = (p_{i\mathtt{a}}, p_{i\mathtt{c}}, p_{i\mathtt{g}}, p_{i\mathtt{t}})$, which motivates the name PPM.

There are many examples in the literature where it is suggested that the independence assumption might be inappropriate and where alternative models are proposed [46, 47]. Still, PPMs remain a very attractive model due to their interpretability and convenience in computational approaches. Moreover, it is not clear whether more advanced approaches bring substantial benefits in modeling transcription factor sequence specificity [48].

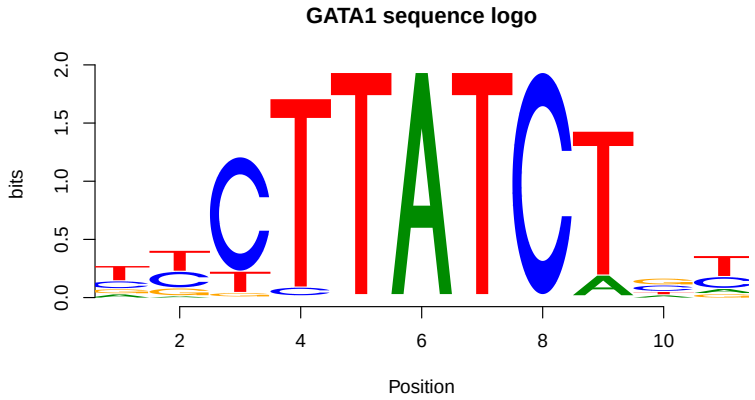

Figure 2.1: *Graphical representation of a PPM via a sequence logo. This example shows the sequence preferences of the transcription factor GATA1 in mouse cells as reported by the JASPAR database [49]. The horizontal axis represents the different positions of the sequence. The height of each symbol is indicative of its frequency.*

PPMs can be represented graphically with sequence logos (see Figure 2.1). For each of the n positions of the PPM, the logo has one column with a height proportional to how informative the position is. More precisely, the height of column i, denoted C_i, is the maximum entropy (2, in this case) minus the actual entropy of the random variable S_i and it is measured in bits:

$$C_i = 2 + \sum_{\alpha \in \{\mathtt{a},\mathtt{c},\mathtt{g},\mathtt{t}\}} p_{i\alpha} \log p_{i\alpha}.$$

As a consequence, if at position i all symbols are equally likely, then the i-th column has height 0, while if only one symbol can appear, the column has height 2.

Additionally, the logo shows which symbols are most likely at each position. These, in fact, are drawn large and at the top of the column. More precisely, the height of symbol α in column i is given by $p_{i\alpha}C_i$ and the symbols are vertically arranged according to their frequency.

In the literature the acronym PPM is not nearly as common as the acronym PWM, which stands for position weight matrix. The difference between the two is that PPMs describe a probabilistic model, while PWMs are a scoring method, typically used to discern the sequences of interest from the rest. PWMs can be derived as the log-likelihood ratio between two PPMs. Denoting with $\boldsymbol{S}^{(1)}$ and $\boldsymbol{S}^{(2)}$ two random sequences modeled as PPMs with coefficients $\boldsymbol{p}^{(1)}$ and $\boldsymbol{p}^{(2)}$, respectively, the corresponding PWM assigns to a sequence $\boldsymbol{s}$ the score:

$$\text{score}(\boldsymbol{s}) = \log \mathrm{P}\left\{\boldsymbol{S}^{(1)} = \boldsymbol{s}\right\} - \log \mathrm{P}\left\{\boldsymbol{S}^{(2)} = \boldsymbol{s}\right\} = \sum_{i=1}^{n} \sum_{\alpha \in \{\mathtt{a},\mathtt{c},\mathtt{g},\mathtt{t}\}} w_{i\alpha} \cdot [s_i = \alpha],$$

where $w_{i\alpha} = \log p_{i\alpha}^{(1)} - \log p_{i\alpha}^{(2)}$ are the weights characterizing the PWM. PWMs find extensive application in the computational prediction of TF binding sites, where one of the two PPMs describes the sequences recognized by the TF, and the other describes the background sequence [15, 45].

2.2.2 The negative binomial distribution

Most of the problems discussed in this work deal with read counts, rather than symbols. While symbols can be modeled with a categorical distribution, a possible model for read counts is the **negative binomial** distribution.

Let X denote a random variable that follows a negative binomial distribution with a mean parameter $\mu \geq 0$ and a size parameter $r > 0$: $X \sim \mathrm{NB}(\mu, r)$. The probability of observing a non-negative integer x is:

$$\mathrm{P}\{X = x\} = \frac{\Gamma(r+x)}{\Gamma(r)x!} \left(\frac{\mu}{r+\mu}\right)^x \left(\frac{r}{r+\mu}\right)^r,$$

where Γ denotes the gamma function. Note that the same distribution can be parametrized differently. A common alternative is to use the parameter pair p and r, where $p = (r+\mu)^{-1}\mu$ is called success probability and r in this context is often called

number of trials, or the parameter pair μ and α, where $\alpha = r^{-1}$ can be referred to as the dispersion parameter. The probability mass function of the negative binomial generalizes that of the Poisson and the geometric distributions, which occur when $r \to \infty$ and when $r = 1$, respectively, and it is therefore a versatile model.

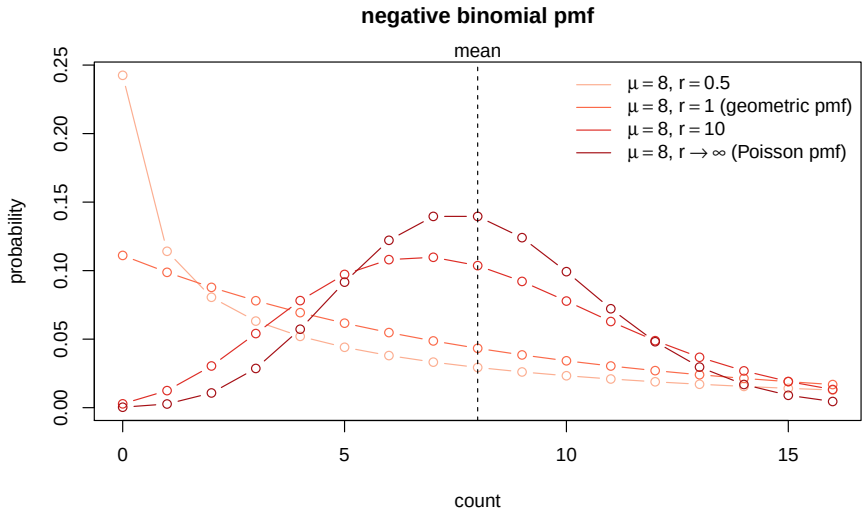

Figure 2.2: *Probability mass function of the negative binomial distribution. All distributions have the same mean (shown by the dashed line). Finite values of r lead to an overdispersed distribution (i.e. a higher variance than the case $r \to \infty$).*

The first two moments are:

$$\mathrm{E}[X] = \mu,$$
$$\mathrm{Var}(X) = \mu + \frac{\mu^2}{r}.$$

This suggests a clear interpretation for the parameters. The parameter μ controls the mean, while r influences the variance relative to the mean.

A fundamental feature is that this distribution can account for **overdispersion**, meaning that the variance can be considerably larger than the mean. This property is frequently observed in count data, which explains why the negative binomial has found extensive application in bioinformatics [50–54].

2.2.3 The negative multinomial distribution

The patterns considered in the next chapters can be represented as sequences of read counts. These can be modeled by a negative multinomial distribution, which was previously used in bioinformatics for modeling footprints in DNase I hypersensitivity data [55].

Let $\boldsymbol{X} = (X_1, X_2, ...X_n)$ denote a random vector that follows a negative multinomial distribution with parameters μ, r and $\boldsymbol{p} = (p_1, p_2, ...p_n)$, where all parameters

are positive and $\sum_{i=1}^{n} p_i = 1$. This can be simply written $\boldsymbol{X} \sim \mathrm{NM}(\mu, r, \boldsymbol{p})$. Sampling from this distribution is equivalent to the following procedure:

1. sample an integer x_+ from a negative binomial distribution with parameters μ and r,

2. sample the integers $(x_1, x_2, ...x_n)$ from a multinomial distribution with x_+ trials and probabilities $p_1, p_2, ...p_n$.

Translating this into formulas, the probability of observing the vector $\boldsymbol{x} = (x_1, x_2, ...x_n)$ is:

$$\begin{aligned} \mathrm{P}\left\{\boldsymbol{X} = \boldsymbol{x}\right\} &= \mathrm{P}\left\{X_+ = x_+\right\} \mathrm{P}\left\{\boldsymbol{X} = \boldsymbol{x} | X_+ = x_+\right\} \\ &= \left\{\frac{\Gamma(r + x_+)}{\Gamma(r) x_+!} \left(\frac{\mu}{r + \mu}\right)^{x_+} \left(\frac{r}{r + \mu}\right)^{r}\right\} \left\{x_+! \prod_{i=1}^{n} \frac{p_i^{x_i}}{x_i!}\right\}, \end{aligned} \tag{2.1}$$

where $x_+ = \sum_{i=1}^{n} x_i$. This distribution can be considered the multivariate generalization of the negative binomial. In fact, by construction $\sum_{i=1}^{n} X_i \sim \mathrm{NB}(\mu, r)$. Moreover, the marginal distribution for each component i is also a negative binomial:

$$X_i \sim \mathrm{NB}(\mu p_i, r).$$

When $r \to \infty$, the negative multinomial converges to the product of n Poisson distributions.

The first two moments are:

$$\begin{aligned} \mathrm{E}[X_i] &= \mu p_i, \\ \mathrm{Cov}(X_i, X_j) &= \frac{\mu^2 p_i p_j}{r} + \mu p_i \cdot [i = j]. \end{aligned}$$

These relations, together with Figure 2.3, illustrate two important properties. First, the marginal distribution of each component is overdispersed. Second, the components are not independent, but they are positively correlated. As it will be shown in the next chapters, both properties are observed in count signals. Another important practical advantage, thoroughly discussed in Chapter 4, is that the parameters can be efficiently learned from the data.

However, the negative multinomial also has some limitations. First, it cannot model component-specific dispersion. That means, for instance, that there is no choice of the r parameter which yields a Poisson-distributed component X_i and at the same time a component X_j that follows a geometric distribution. Second, the overdispersion and correlation properties are linked by a single parameter r, which means that it is not possible to model a large overdispersion with a weak correlation between the components, or a weak overdispersion with a large correlation.

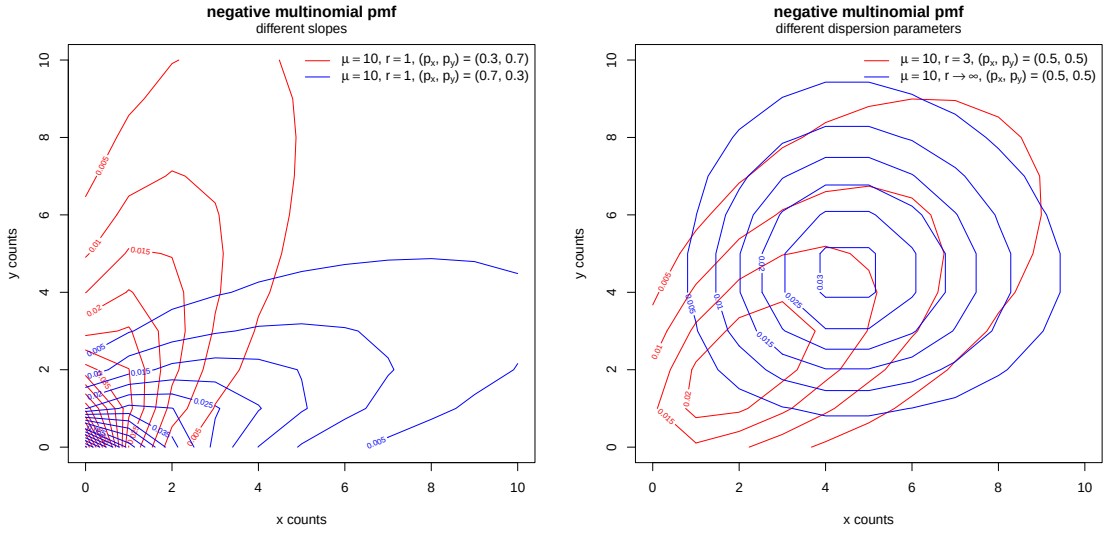

Figure 2.3: *Probability density function of the negative multinomial distribution. On the left, the joint probability mass function of a bivariate negative multinomial distribution with different values of p_x and p_y. The counts are not independent, but they are preferentially distributed along a line and therefore positively correlated. The p_x and p_y parameters control the slope of this line. On the right, influence of the size parameter r. When $r \to \infty$ the marginal distributions are uncorrelated and independent, while with finite values of r the correlation and the dispersion increase.*

2.2.4 Maximum likelihood estimation

The same way probability distributions can characterize patterns, patterns can characterize probability distributions. Let X denote a random variable with a probability distribution that depends on a parameter θ, so that the probability of observing x can be written as $\mathrm{P}\{X = x; \theta\}$. On one hand, the function $f_\theta(x) = \mathrm{P}\{X = x; \theta\}$ specifies how probable it is to observe x given the parameter θ. On the other hand, the function $f_x(\theta) = \mathrm{P}\{X = x; \theta\}$, called the likelihood function, can suggest how appropriate the parameter θ is given the observed data. The optimal parameter $\theta^{(\mathrm{opt})}$ can then be chosen as:

$$\theta^{(\mathrm{opt})} = \arg\max_\theta \mathrm{P}\{X = x; \theta\}\,. \tag{2.2}$$

This criterion is called maximum likelihood estimation (MLE), and it is one of the basic pillars of statistical inference (see for instance Casella & Berger [56]). Often the likelihood is computed from n observations $x_1, x_2, ..., x_n$ that are assumed to be drawn from independent and identically distributed (iid) random variables $X_1, X_2, ...X_n$. In this case the maximum likelihood equation can be written as:

$$\theta^{(\mathrm{opt})} = \arg\max_\theta \sum_{i=1}^{n} \log \mathrm{P}\{X_i = x_i; \theta\}\,. \tag{2.3}$$

Note that MLE is not the only possible criterion for parameter inference. When the set of observations $\boldsymbol{x}$ is small, for instance, it is usually better to incorporate additional knowledge about the parameter θ in the form of a prior distribution and to use maximum a posteriori estimation (MAP) or bayesian inference [57]. In the problems discussed in this thesis, however, the set of observations is large and MLE will be used.

2.2.5 Expectation maximization

In some cases, Equation 2.2 is difficult to solve with standard numerical methods. Expectation maximization (EM) is a technique for performing maximum likelihood estimation in situations where the probability of the observed data $\boldsymbol{x}$ can be expressed as:

$$\mathrm{P}\left\{\boldsymbol{X}=\boldsymbol{x};\theta\right\}=\sum_{\boldsymbol{z}\in S_{\boldsymbol{Z}}}\mathrm{P}\left\{\boldsymbol{X}=\boldsymbol{x},\boldsymbol{Z}=\boldsymbol{z};\theta\right\}, \tag{2.4}$$

where the random vector $\boldsymbol{X}$ represents the observed variables and the random vector $\boldsymbol{Z}$, taking values in the set $S_{\boldsymbol{Z}}$, represents the hidden variables.

EM is an iterative procedure that starts from an initial arbitrary choice of the parameter $\theta^{(0)}$ and produces increasingly "better" estimates $\theta^{(1)},\theta^{(2)},...$ until no improvement in the likelihood function is observed. At each iteration t, the parameters are updated according to the formula:

$$\theta^{(t+1)}=\arg\max_{\theta}\sum_{\boldsymbol{z}\in S_{\boldsymbol{Z}}}\mathrm{P}\left\{\boldsymbol{Z}=\boldsymbol{z}|\boldsymbol{X}=\boldsymbol{x};\theta^{(t)}\right\}\log\mathrm{P}\left\{\boldsymbol{X}=\boldsymbol{x},\boldsymbol{Z}=\boldsymbol{z};\theta\right\}. \tag{2.5}$$

The objective function in this maximization problem can be interpreted as the expected log-likelihood of the observed data, where each $\boldsymbol{z}$ is weighted by its conditional probability given $\boldsymbol{x}$ and $\theta^{(t)}$. This interpretation motivates the name expectation maximization.

It can be easily shown that this update rule yields a sequence of parameters $\theta^{(t)}$ such that $\mathrm{P}\left\{\boldsymbol{X}=\boldsymbol{x};\theta^{(t)}\right\}\leq\mathrm{P}\left\{\boldsymbol{X}=\boldsymbol{x};\theta^{(t+1)}\right\}$ (see, for instance, Little & Rubin [58]). The sequence typically stops when there is no further increase in the objective function and the algorithm is said to converge. The major drawback of this approach is that the final parameter of the sequence is not guaranteed to be a global optimum and that different initial parameters can lead to different final parameters. Mixture models and hidden Markov models are typical frameworks where EM can be convenient.

2.2.6 Mixture models

Mixture models are a natural way of formulating a clustering problem in probabilistic terms (see, for instance, gaussian mixture models (Reynolds [59] and Figure 2.4)). In a clustering problem n datapoints $\boldsymbol{x}=(x_1,x_2,...,x_n)$ are given and they are assumed to belong to k different families, called clusters. A clustering method, therefore, needs to identify these families and to assign each datapoint to a cluster.

In a mixture model, it is assumed that each of the n datapoints is generated by one among k random variables. Each cluster, therefore, is characterized by a random variable.

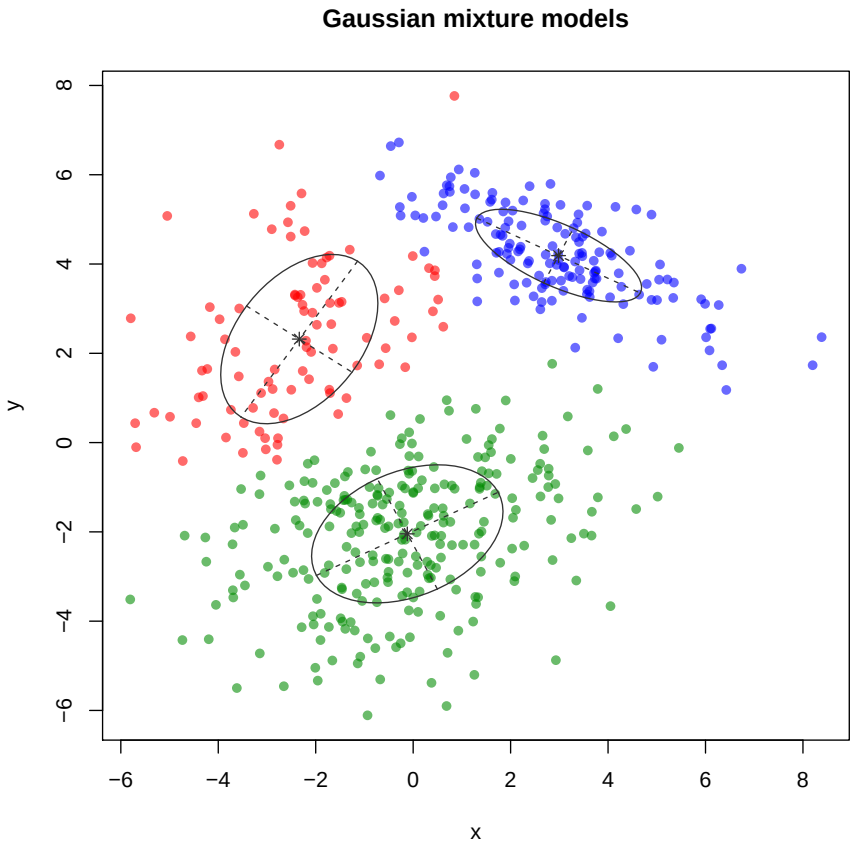

Figure 2.4: *Clustering with gaussian mixture models. The figure shows a set of observations consisting of an x and y values and represented as solid circles. The gaussian mixture model framework has been applied to separate the datapoints into different clusters. By maximizing the likelihood of this probabilistic model it is possible to (i) assign a cluster to each observation (represented by the color of the circle) and (ii) characterize each cluster in probabilistic terms. In this example each cluster is characterized by a multivariate gaussian distribution, which is represented by ellipsis.*

Formally, let $\boldsymbol{x} = (x_1, x_2, ..., x_n)$ denote a set of n observations and let $M_1, M_2, ...M_k$ denote k random variables with parameters $\boldsymbol{\nu} = (\nu_1, \nu_2, ..., \nu_k)$. Each cluster has a relative frequency specified by the mixing coefficients $\boldsymbol{\pi} = (\pi_1, \pi_2, ...\pi_k)$, which are positive probabilities that sum to 1. The observations x_i are assumed to be iid and generated using the following procedure:

1. choose one of the random variables M_j by sampling j according to the mixing coefficients $\boldsymbol{\pi}$,

2. draw an observation from the random variable M_j, which is controlled by the parameter ν_j.

In formulas, the joint probability of choosing the random variables with indices $\boldsymbol{z} = (z_1, z_2, ..., z_n)$ and generating the data $\boldsymbol{x}$ is:

$$\begin{aligned} \mathrm{P}\left\{\boldsymbol{X} = \boldsymbol{x}, \boldsymbol{Z} = \boldsymbol{z}\right\} &= \prod_{i=1}^{n} \mathrm{P}\left\{X_i = x_i | Z_i = z_i\right\} \mathrm{P}\left\{Z_i = z_i\right\} \\ &= \prod_{i=1}^{n} \mathrm{P}\left\{M_{z_i} = x_i\right\} \pi_{z_i}. \end{aligned} \tag{2.6}$$

If all the parameters of the model are given, it is straightforward to infer the index z_i of the random variable that most likely generated the datapoint i:

$$z_i = \underset{j \in \{1,...,k\}}{\arg\max} \mathrm{P}\left\{Z_i = j | \boldsymbol{X} = \boldsymbol{x}\right\},$$

where

$$\mathrm{P}\left\{Z_i = j | \boldsymbol{X} = \boldsymbol{x}\right\} = \mathrm{P}\left\{Z_i = j | X_i = x_i\right\} = \frac{\mathrm{P}\left\{M_j = x_i\right\} \pi_j}{\sum_{h=1}^{k} \mathrm{P}\left\{M_h = x_i\right\} \pi_h}. \tag{2.7}$$

The coefficients $\mathrm{P}\left\{Z_i = j | \boldsymbol{X} = \boldsymbol{x}\right\}$ are called posterior probabilities, and they play an important role also in learning (see next subsection).

2.2.7 EM for mixture models

Learning the parameters $\boldsymbol{\nu}$ and $\boldsymbol{\pi}$ from the data $\boldsymbol{x}$ using MLE amounts to maximizing the likelihood function:

$$\mathrm{P}\left\{\boldsymbol{X} = \boldsymbol{x}; \boldsymbol{\nu}, \boldsymbol{\pi}\right\} = \sum_{\boldsymbol{z} \in \{1,...,k\}^n} \mathrm{P}\left\{\boldsymbol{X} = \boldsymbol{x}, \boldsymbol{Z} = \boldsymbol{z}; \boldsymbol{\nu}, \boldsymbol{\pi}\right\}.$$

This is equivalent to Equation 2.4 if we view the parameter θ as comprising the parameters $\boldsymbol{\nu}$ and $\boldsymbol{\pi}$. The expectation maximization technique can therefore be applied to maximize the likelihood of a mixture model. We derive now update rules for the parameters $\boldsymbol{\nu}$ and $\boldsymbol{\pi}$ that will be used in the next chapters.

Let $\theta^{(t)} = (\boldsymbol{\nu}^{(t)}, \boldsymbol{\pi}^{(t)})$ denote the parameter estimates at the end of the t-th iteration. Using Equation 2.6, the objective function in Equation 2.5 can be rewritten as:

$$\begin{aligned} &\sum_{\boldsymbol{z} \in \{1,...,k\}^n} \mathrm{P}\left\{\boldsymbol{Z} = \boldsymbol{z} | \boldsymbol{X} = \boldsymbol{x}; \theta^{(t)}\right\} \log \left\{ \prod_{i=1}^{n} \mathrm{P}\left\{M_{z_i} = x_i; \nu_{z_i}\right\} \pi_{z_i} \right\} \\ = &\sum_{i=1}^{n} \sum_{\boldsymbol{z} \in \{1,...,k\}^n} \mathrm{P}\left\{\boldsymbol{Z} = \boldsymbol{z} | \boldsymbol{X} = \boldsymbol{x}; \theta^{(t)}\right\} \log \left\{\mathrm{P}\left\{M_{z_i} = x_i; \nu_{z_i}\right\} \pi_{z_i}\right\}. \end{aligned}$$

Noting that $\{1,...,k\}^n = \bigcup_{j=1}^{k} (\{1,...,k\}^{i-1} \times \{j\} \times \{1,...,k\}^{n-i})$, the last term can be rewritten as:

$$\sum_{i=1}^{n}\sum_{j=1}^{k}\sum_{\boldsymbol{z}:z_i=j} \mathrm{P}\left\{\boldsymbol{Z}=\boldsymbol{z}|\boldsymbol{X}=\boldsymbol{x};\theta^{(t)}\right\}\log\left\{\mathrm{P}\left\{M_j=x_i;\nu_j\right\}\pi_j\right\} =$$
$$\sum_{i=1}^{n}\sum_{j=1}^{k}\mathrm{P}\left\{Z_i=j|\boldsymbol{X}=\boldsymbol{x};\theta^{(t)}\right\}\log\left\{\mathrm{P}\left\{M_j=x_i;\nu_j\right\}\pi_j\right\}.$$

The coefficients $\mathrm{P}\left\{Z_i=j|\boldsymbol{X}=\boldsymbol{x};\theta^{(t)}\right\} = \mathrm{P}\left\{Z_i=j|X_i=x_i;\theta^{(t)}\right\}$ are the previously mentioned posterior probabilities and they can be easily computed at each iteration t from Equation 2.7. Denoting

$$\gamma_{ij}^{(t)} = \mathrm{P}\left\{Z_i=j|\boldsymbol{X}=\boldsymbol{x};\theta^{(t)}\right\},$$

the update rule becomes simply:

$$\theta^{(t+1)} = \arg\max_{\theta}\left\{\left\{\sum_{j=1}^{k}\sum_{i=1}^{n}\gamma_{ij}^{(t)}\log\pi_j\right\} + \left\{\sum_{j=1}^{k}\sum_{i=1}^{n}\gamma_{ij}^{(t)}\log\mathrm{P}\left\{M_j=x_i;\nu_j\right\}\right\}\right\}.$$

This allows to maximize each parameter set independently:

$$\pi_j^{(t+1)} = \left(\sum_{i=1}^{n}\sum_{h=1}^{k}\gamma_{ih}^{(t)}\right)^{-1}\sum_{i=1}^{n}\gamma_{ij}^{(t)},$$
$$\nu_j^{(t+1)} = \arg\max_{\nu}\sum_{i=1}^{n}\gamma_{ij}^{(t)}\log\mathrm{P}\left\{M_j=x_i;\nu\right\}, \tag{2.8}$$

where the first rule follows from Lemma B.1.2. The exact way the parameters ν_j are updated depends on the probabilistic assumptions on the random variables M_j. Note also how Equation 2.8 can be considered a weighted maximum likelihood estimation problem and a generalization of Equation 2.3. Once a method for solving this problem has been implemented, the mixture models assumptions can be immediately applied to obtain a probabilistic clustering algorithm. This is why we often refer to mixture models as a framework, where additional modules can be plugged in to completely specify the algorithm. We will see two different applications of this framework in Chapter 3 and 4.

2.2.8 Hidden Markov models

Hidden Markov models (HMMs) are a probabilistic framework used to segment a sequence of observations. Differently than a clustering method, a segmentation method does not assign a class to each observation independently, but assumes a certain dependence between adjacent observations. The classes predicted by an HMM can be considered a sort of smoothed clustering, because they tend to form segments, i.e. runs of adjacent observations assigned to the same class. HMMs have

been widely applied in many different fields, most notably in signal processing [60]. They have also become a standard tool in bioinformatics, in particular for modeling patterns in biological sequences [61].

For ease of discussion we will assume that only one sequence of observations X is given, even though, in general, HMMs are applied to more than one sequence. The main assumption of the model is that there are k possible states and that the sequence of n observations $\boldsymbol{x} = (x_1, x_2, ..., x_n)$ is explained by a hidden sequence of n states $\boldsymbol{z} = (z_1, z_2, ..., z_n)$.

Let the random variables Z_i and X_i denote the hidden state and the random observation at position i, respectively. The complete set of model parameters consists of the parameters $\boldsymbol{\pi}$, called **initial probabilities** and characterizing the first hidden state of the sequence, $\boldsymbol{\alpha}$ called **transition probabilities** and specifying the transitions between hidden states, and $\boldsymbol{\nu}$, that we will call emission parameters, describing what symbols are expected to be observed from a given state. It is assumed that the sequence of observations $\boldsymbol{X} = (X_1, X_2, ..., X_n)$ is generated by the following procedure:

1. The first hidden state Z_i is chosen by sampling according to the initial probabilities $\boldsymbol{\pi} = (\pi_1, \pi_2, ...\pi_k)$,

2. Given that the current state is j, an observation X_i is generated by sampling from the random variable M_j with parameter ν_j,

3. Given that the current state is j, the next state is chosen by sampling according to the transition probabilities $\boldsymbol{\alpha}_j = (\alpha_{j1}, \alpha_{j2}, ...\alpha_{jk})$, which constitute the j-th row of the square matrix $\boldsymbol{\alpha} = (\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, ..., \boldsymbol{\alpha}_k)$.

Translating this description into formulas, the joint probability of the data $\boldsymbol{x}$ and the hidden states $\boldsymbol{z}$ is:

$$\mathrm{P}\left\{\boldsymbol{X} = \boldsymbol{x}, \boldsymbol{Z} = \boldsymbol{z}\right\} = \pi_{z_1} \mathrm{P}\left\{M_{z_1} = x_1\right\} \prod_{i=2}^{n} \alpha_{z_{i-1} z_i} \mathrm{P}\left\{M_{z_i} = x_i\right\}. \tag{2.9}$$

Suppose that we know the parameters and we observe the sequence $\boldsymbol{x}$, how do we infer which state generated each observation? This problem is called decoding. There are two common approaches to this problem. The first, called Viterbi decoding, consists in finding the state sequence $\boldsymbol{z}^{(V)}$ that maximizes the probability given the data:

$$\boldsymbol{z}^{(\mathrm{V})} = \underset{\boldsymbol{z} \in \{1,...,k\}^n}{\arg\max}\ \mathrm{P}\left\{\boldsymbol{Z} = \boldsymbol{z} | \boldsymbol{X} = \boldsymbol{x}\right\}.$$

The hidden state for observation i is then simply $z_i^{(\mathrm{V})}$. The second method, called posterior decoding, uses a similar criterion, but for each state separately:

$$z_i^{(\mathrm{P})} = \underset{j \in \{1,...,k\}}{\arg\max}\, \mathrm{P}\left\{Z_i = j | \boldsymbol{X} = \boldsymbol{x}\right\}.$$

As in the mixture models, the probabilities $\mathrm{P}\left\{Z_i = j | \boldsymbol{X} = \boldsymbol{x}\right\}$ are called posterior probabilities and will be denoted by γ_{ij}. The state sequence resulting from the

posterior decoding $(z_1^{(\mathrm{P})}, z_2^{(\mathrm{P})}, ..., z_n^{(\mathrm{P})})$ might be very improbable compared to others, but the predicted state for any position i is, considering all possible paths, the most probable. The algorithms that perform the Viterbi and posterior decoding are called the Viterbi and forward-backward algorithm, respectively, but they will not be described here (see [61] for a description).

2.2.9 EM for hidden Markov models

Often HMMs are used in contexts where the only given parameter is the number of hidden states. All other parameters are unknown and need to be estimated solely from the observations (unsupervised learning). This can be done by maximizing the likelihood of the data:

$$\mathrm{P}\left\{\boldsymbol{X} = \boldsymbol{x}; \boldsymbol{\nu}, \boldsymbol{\alpha}, \boldsymbol{\pi}\right\} = \sum_{\boldsymbol{z} \in \{1,...,k\}^n} \mathrm{P}\left\{\boldsymbol{X} = \boldsymbol{x}, \boldsymbol{Z} = \boldsymbol{z}; \boldsymbol{\nu}, \boldsymbol{\alpha}, \boldsymbol{\pi}\right\}.$$

As for the mixture models, the application of EM yields an iterative optimization algorithm for the parameters of the HMM, which is typically referred to as the Baum-Welch algorithm. In the following, the parameter θ represents the full parameter set $(\boldsymbol{\nu}, \boldsymbol{\alpha}, \boldsymbol{\pi})$. After using Equation 2.9, the optimization function in Equation 2.5 becomes

$$\sum_{\boldsymbol{z} \in \{1,...,k\}^n} \mathrm{P}\left\{\boldsymbol{Z} = \boldsymbol{z} | \boldsymbol{X} = \boldsymbol{x}; \theta^{(t)}\right\} \left\{ \sum_{i=1}^{n} \log \mathrm{P}\left\{M_{z_i} = x_i; \nu_{z_i}\right\} + \sum_{i=2}^{n} \log \alpha_{z_{i-1} z_i} + \log \pi_{z_1} \right\}.$$

Proceeding in a similar way as for the mixture models, the last expression can be transformed into

$$\sum_{i=1}^{n} \sum_{j=1}^{k} \gamma_{ij}^{(t)} \log \mathrm{P}\left\{M_j = x_i; \nu_j\right\} + \sum_{i=2}^{n} \sum_{u=1}^{k} \sum_{v=1}^{k} \xi_{iuv}^{(t)} \log \alpha_{uv} + \sum_{j=1}^{k} \gamma_{1j}^{(t)} \log \pi_j,$$

where

$$\begin{aligned} \gamma_{ij}^{(t)} &= \mathrm{P}\left\{Z_i = j | \boldsymbol{X} = \boldsymbol{x}; \theta^{(t)}\right\}, \\ \xi_{iuv}^{(t)} &= \mathrm{P}\left\{Z_{i-1} = u, Z_i = v | \boldsymbol{X} = \boldsymbol{x}; \theta^{(t)}\right\}. \end{aligned}$$

These coefficients can be computed with the previously mentioned forward-backward algorithm. Now each set of parameters can be maximized independently, which yields the update rules:

$$\begin{aligned} \pi_j^{(t+1)} &= \gamma_{1j}^{(t)}, \\ \alpha_{uv}^{(t+1)} &= \left(\sum_{i=1}^{n} \sum_{w=1}^{k} \xi_{iuw}^{(t)} \right)^{-1} \sum_{i=1}^{n} \xi_{iuv}^{(t)}, \\ \nu_j^{(t+1)} &= \arg\max_{\nu} \sum_{i=1}^{n} \gamma_{ij}^{(t)} \log \mathrm{P}\left\{M_j = x_i; \nu\right\}, \end{aligned} \tag{2.10}$$

where the first two rules follow from Lemma B.1.2. The last rule is identical to the one in Equation 2.8, meaning that these two probabilistic frameworks are very similar from a computational point of view. In Chapter 4 we will see an algorithm that solves this optimization problem for the negative multinomial distribution and which is used to perform unsupervised learning both with mixture models and hidden Markov models.

Chapter 3

Nucleosome detection

This chapter presents NucHunter: an algorithm that uses the data from ChIP-seq experiments directed against many histone modifications to infer positioned nucleosomes and to annotate each nucleosome with the intensities of the histone modifications.

3.1 Motivation

Nucleosomes are the basic repeating units of eukaryotic chromatin. Each nucleosome is a complex formed by 8 proteins, called the core histones, and a stretch of about 147 bps of DNA wound around the histone octamer [62]. Nucleosomes and histones have been related to many regulatory functions [23, 63], but their role is not yet fully understood and constitutes an active area of research [64] (see also Subsection 1.1.6). In particular, to study how nucleosomes interact with other molecular processes and the factors guiding nucleosome formation and eviction, it is important to detect nucleosomes' positions along the genome.

In the last years MNase-seq has been the method of choice for obtaining genome-wide nucleosome maps, especially in yeast [65–68]. Briefly, crosslinked chromatin is digested with micrococcal nuclease, an enzyme with endo- and exonuclease activity, so as to obtain mononucleosomal DNA fragments. These fragments are then purified and sequenced and the resulting reads, which precisely map to the borders of nucleosome formation sites, are used to infer nucleosome positions. However, today many consortia, such as NIH Roadmap Epigenomics and ENCODE [69, 70], are providing a large number of histone modifications ChIP-seq datasets in human, which are specific for histones with a particular modification (see Subsection 1.2.1). Because the core histone proteins are part of a stable protein-DNA complex, it is natural to assume that the localization of modified histone proteins corresponds to the position of the nucleosomes. This suggests that histone modification ChIP-seq data can be used to infer nucleosome positions.

However, this is far from being a simple task. First, because the nucleosome signal in ChIP-seq data is confused by sparse sampling and high noise levels. Second, because the DNA fragments obtained with a ChIP experiment are not as tightly

centered around the nucleosomal DNA as with an MNase-seq experiment. Third, because the degree of positioning can vary considerably across the genome [25]. In some regions, in fact, nucleosomes tend to occupy almost the same location within a homogeneous cell population. These are called well positioned nucleosomes and they occur, for instance, at promoters of active genes. In other regions, by contrast, such as in the body transcribed genes, nucleosomes occupy different positions in different cells and at different times. Using the terminology defined in Landt *et al.* [71], both point-source and broad-source peaks can be detected in ChIP-seq experiments for histones. To obtain a comprehensive and reliable set of predictions, one should combine the information contained in as many distinct ChIP-seq experiments as possible and allow for some plasticity in the shape of the signal.

A number of tools for the inference of nucleosome positions have already been developed. Some of them use segmentation approaches to detect large domains of high nucleosome abundance [72, 73]. These tools, however, are not designed to detect well positioned nucleosomes. Other tools apply signal processing techniques, such as Fourier transforms [74], wavelet decomposition [75] and ad hoc filters [76, 77], to smooth the count signal, followed by the detection of local maxima. Others are based on Bayesian modeling of the nucleosome enrichment pattern [78, 79] and on Monte Carlo simulations [80]. However, these methods do not control for systematic biases by comparing the nucleosome calls with data from control experiments. Furthermore, they cannot integrate data from multiple histone marks in a straightforward manner. Finally, due to the potentially large genome size and the high number of modified nucleosomes, especially in human cells, the runtime of these tools may be prohibitive.

We present NucHunter: an algorithm that uses ChIP-seq data to detect well positioned nucleosomes. NucHunter overcomes the limitations of the available tools, can detect well positioned nucleosomes more accurately, and presents unique features. First, it can use information from a control sample to correct for systematic biases inherent in this high-throughput technology. Second, it is designed to integrate multiple histone marks to broaden the range of nucleosome positions that can be detected. Third, it annotates each identified nucleosome with the contributing histone modifications. We will demonstrate that these annotations can be used to cluster nucleosomes by their histone modification patterns. These clusters can be correlated to the function of the chromatin, such as transcriptional start sites and enhancers, or to the underlying process, such as transcriptional elongation by RNA polymerase II. The results support the assumption that nucleosomes serve as signal modules for biological processes and that the corresponding histone modification patterns are a reflection of the signaling taking place on these modules [81].

3.2 Methods

The algorithm performs a preprocessing step, where the input files, containing the chromosomal positions of mapped reads, are turned into a single count signal, a peak calling step, where candidate positions for nucleosome formation sites are detected,

and additional postprocessing steps, where these candidates are filtered and scored accounting for a number of possible sources of bias.

3.2.1 Preprocessing

Let us first consider a single ChIP-seq experiment. We denote with the symbol c_i the number of reads whose 5' end maps at position i. For simplicity, we will treat i as a simple integer ranging from 1 to the genome length w_{gen}, but in practice i also specifies a chromosome. To distinguish between reads mapping to the positive and negative strand we use positive and negative signs. Hence, c_{+i} and c_{-i}, denote the read counts at position i relative to the positive and negative strand, respectively, and they can be considered elements of a signal $\boldsymbol{c}$.

A well positioned nucleosome typically exhibits a peak of positive strand reads upstream the nucleosome location, and one of negative strand reads downstream. The distance between these two peaks roughly equals the average length of a fragment in the DNA library, which will be denoted by f (see Figure 3.1). To obtain a consensus signal, denoted by $\boldsymbol{d}$, the counts on the positive strand are shifted to the right, those on the negative are shifted to the left, and the sum of the two is considered. The amount of this shift is about $f/2$ (rounded to the closest integer), which yields the consensus signal:

$$d_i = c_{+(i-f/2)} + c_{-(i+f/2)}.$$

Usually the average fragment length needs to be estimated from the data itself. There are some tools that can perform this estimation (such as Zhang *et al.* [16]), however, we found them unsatisfactory when applied to histone marks. Therefore, as part of NucHunter, we also provide a method for estimating the average fragment length (described in Section 3.2.6).

Let us now consider the case where $n_{\text{mark}} > 1$ ChIP-seq samples are used. The procedure above is applied to each sample independently, so that n_{mark} sample-specific signals $\boldsymbol{d}^{(1)}, \boldsymbol{d}^{(2)}, ...\boldsymbol{d}^{(n_{\text{mark}})}$ are obtained. Finally, the consensus signal $\boldsymbol{d}$ is simply the sum of the sample-specific signals. Note that, in general, for different samples different fragment lengths f will be used.

3.2.2 Peak detection

In the peak detection step (see Figure 3.2) the consensus signal is smoothed, then local maxima in the smoothed signal are detected and a threshold is applied to discard background peaks.

Smoothing is performed using a particular filter. A filter (more precisely a linear time-invariant filter) is characterized by a discrete signal $\boldsymbol{k}$ called impulse response. Given a consensus signal $\boldsymbol{d}$, the filter output $\boldsymbol{o}$ is the result of the following operation, called convolution:

$$o_i = (\boldsymbol{d} * \boldsymbol{k})_i = \sum_{j=-\infty}^{\infty} d_{i-j} k_j. \tag{3.1}$$

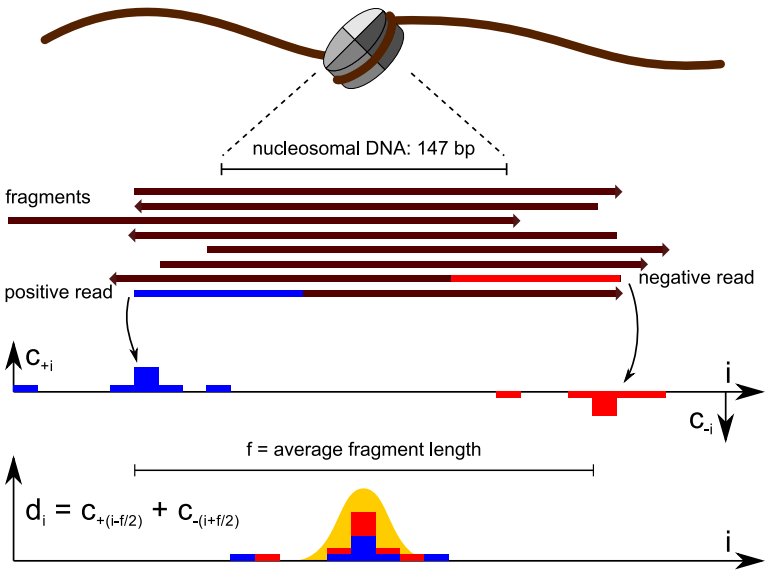

Figure 3.1: *Preprocessing: from DNA fragments to consensus signal. The horizontal axis represents the reference genome. Above, a cartoon of a nucleosome wound around double-stranded DNA. Below, single-stranded DNA fragments relative to the nucleosome above and isolated via immunoprecipitation (the arrow denotes the 3' end). Reads are the 5' portion of a fragment. Reads are counted so that c_{+i} and c_{-i} denote the number of positive and negative of reads whose 5' end maps at position i. The consensus signal is obtained by shifting and summing the strand-specific read counts approximately $f/2$ bases downstream, where f is the average fragment length.*

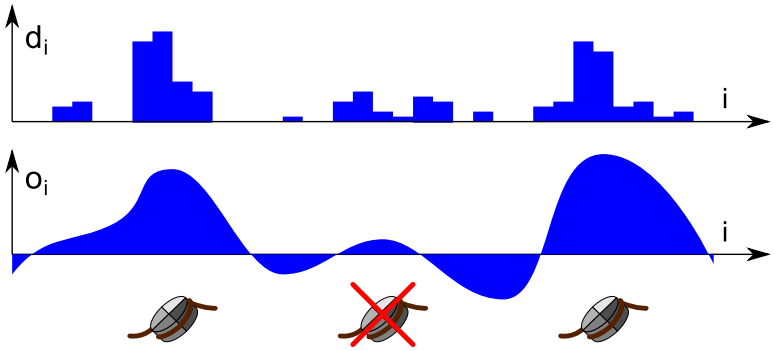

Figure 3.2: *Peak detection from the consensus signal. The consensus signal* $\boldsymbol{d}$ *(shown on top) is smoothed using a linear filter, which yields the filter output* $\boldsymbol{o}$ *(shown on the bottom). The local maxima in the filter output (shown with the symbol of a nucleosome) are detected and the non-significant ones are filtered out.*

In our approach the impulse response has been chosen according to the following two criteria: first, it must separate sharp peaks from more spread out read distributions or non-enriched regions; second, it must have good smoothing properties, so that the convoluted signal contains a limited number of local maxima and, therefore, the algorithm returns fewer false positives. We chose as impulse response the second derivative of a gaussian density function, also known as the Mexican hat wavelet (see Figure 3.3):

$$k_i = (1 - \frac{i^2}{\sigma_{\text{mh}}^2}) \exp\left\{-\frac{i^2}{2\sigma_{\text{mh}}^2}\right\}.$$

The Mexican hat wavelet removes both high- and low-frequency components (band-pass filter) from the Fourier spectrum of the consensus signal. This is appropriate in our case, where we interpret high frequencies as random oscillations due to noise or insufficient coverage, and low frequencies as ambiguous domains due to broad-source peaks or local biases (such as GC content or open chromatin [82]). The wavelet is parametrized by the scale parameter σ_{mh}, which is an input parameter. In our studies, we set σ_{mh} to 50 because, in general, it is a good compromise between calling too many peaks and merging closely spaced ones.

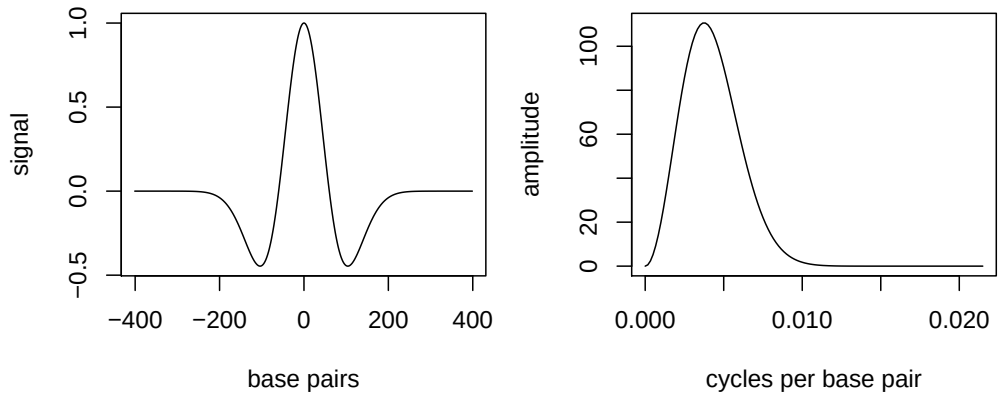

Figure 3.3: *The Mexican hat wavelet. The plots on the left and right show respectively the signal and its frequency spectrum for* $\sigma_{mh} = 50$.

Obtaining the convoluted signal for large genomes poses computational problems. In fact, using a naïve approach, a long signal as impulse response results in a slow convolution. In NucHunter the convolution has been implemented using recursive gaussian filters, which yields a runtime linear in the length of the consensus signal and independent on σ_{mh} [83–85].

Once local maxima are extracted from the filter output, some of them are discarded based on a simple statistical method. We model the noise by assuming that values of the consensus signal $\boldsymbol{d}$ are realizations of independent and identically distributed random variables $\boldsymbol{D}$. Using this assumption, we derive the mean and standard deviation of the convoluted signal $\boldsymbol{o}$, and we assign a z-score to each local maximum. Finally, peaks are discarded based on a threshold on the z-score. Let O_i

denote the random variable that generates the value o_i. The random vector $\boldsymbol{O}$ can be expressed as a function of $\boldsymbol{D}$ using the convolution formula 3.1:

$$O_i = \sum_{j=-\infty}^{\infty} D_{i-j} k_j.$$

Because the elements in $\boldsymbol{D}$ are assumed to be iid, the mean and standard deviation of O_i are:

$$\mathrm{E}[O_i] = \mathrm{E}[D_i] \sum_{j=-\infty}^{\infty} k_j,$$

$$\mathrm{Var}(O_i) = \mathrm{Var}(D_i) \sum_{j=-\infty}^{\infty} k_j^2.$$

$\mathrm{E}[D_i]$ and $\mathrm{Var}(D_i)$, which do not depend on i, are estimated by computing the sample mean and sample variance of $\boldsymbol{d}$. Finally, the z-score z_i associated to the value o_i is:

$$z_i = \frac{o_i - \mathrm{E}[O_i]}{\sqrt{\mathrm{Var}(O_i)}}.$$

The detected peaks are all those local maxima with a z-score above a certain threshold. This z-score represents the strength of a peak, and a user-defined threshold, whose default value is 3, specifies how many standard deviations above average the peaks' strength must be.

Note that, in general, it is unrealistic to assume that the elements in $\boldsymbol{D}$ are iid, because many biases, such as GC content, affect multiple adjacent base pairs simultaneously. However, the purpose of this procedure is to standardize the values of the output signal so that an interpretable threshold can be applied, rather than to perform an accurate statistical test.

3.2.3 Postprocessing

After a set of putative peaks has been derived, additional steps are applied to filter and annotate them (note that the world filter, from now on, should not be confused with the signal processing technique outlined in Subsection 3.2.2). All these steps are based on the enrichment level of a peak. Let p denote the peak's position. The enrichment level e_p is the total number of reads that contribute to the consensus signal in a window of a certain radius w_{enr} (which defaults to 73):

$$e_p = \sum_{j=-w_{\mathrm{enr}}}^{w_{\mathrm{enr}}} d_{p+j}.$$

The first filtering step consists in controlling the relative amount of positive and negative reads in the enrichment level. In fact, highly unbalanced contributions from the two strands are difficult to interpret and likely to arise from mapping

artifacts. Following the approach of Zhang *et al.* [75], we filter out peaks where the enrichment level from one strand is four times larger or four times smaller than the enrichment level from the other. Next, in case a control sample is available, peaks are filtered based on the significance of the enrichment level and, in case multiple ChIP-seq experiments have been used, peaks are annotated with the histone marks active on them.

3.2.4 Enrichment test

Given a control sample, peaks are filtered in a similar manner as in Zhang *et al.* [16]: the enrichment level is modeled as a Poisson random variable whose parameter is estimated from both a global and a local average of the control sample. From this model a p-value is obtained and peaks are filtered based on a p-value threshold.

In more detail, let $\overline{\boldsymbol{d}}$ represent the control signal, preprocessed from the control reads in the same way as the signal $\boldsymbol{d}$. To make the two experiments comparable, a scaling factor α is computed that takes into account the differences in sequencing coverage:

$$\alpha = \frac{\sum_{i=1}^{w_{\text{gen}}} d_i}{\sum_{i=1}^{w_{\text{gen}}} \overline{d}_i}.$$

The local and global noise estimates per base pair $\lambda_p^{(\text{loc})}$ and $\lambda^{(\text{glob})}$ are computed as:

$$\lambda_p^{(\text{loc})} = \alpha \frac{\sum_{j=-w_{\text{loc}}}^{w_{\text{loc}}} \overline{d}_{p+j}}{2w_{\text{loc}} + 1}, \quad \lambda^{(\text{glob})} = \alpha \frac{\sum_{i=1}^{w_{\text{gen}}} \overline{d}_p}{w_{\text{gen}}},$$

where w_{loc} is 1000 base pairs by default. The noise estimate for the read counts at position p, denoted λ_p, is chosen as the maximum between the local and global estimate: $\lambda_p = \max\{\lambda_p^{(\text{loc})}, \lambda^{(\text{glob})}\}$. Finally, the random variable E_p representing the null model for the enrichment level e_p follows a Poisson distribution:

$$E_p \sim \text{Pois}((2w_{\text{enr}} + 1)\lambda_p).$$

This allows to compute a p-value for each peak and to apply a user-defined threshold, which defaults to 10^{-5}, to discard non-significant peaks.

3.2.5 Histone mark annotation

A final step takes place when the sample is obtained from multiple ChIP-seq experiments and a control sample is available. The enrichment level at each peak is decomposed into the contributions from the different samples and these are compared to the noise estimate (see Figure 3.4). More precisely, the same procedure outlined in Subsection 3.2.4, which compares the consensus signal $\boldsymbol{d}$ to the control signal $\overline{\boldsymbol{d}}$ at the candidate peaks' locations, is now applied using the sample-specific signals $\boldsymbol{d}^{(1)}, \boldsymbol{d}^{(2)}, ...\boldsymbol{d}^{(n_{\text{mark}})}$ in place of $\boldsymbol{d}$. If the enrichment level of a mark at a peak is not significant, the mark is considered inactive.

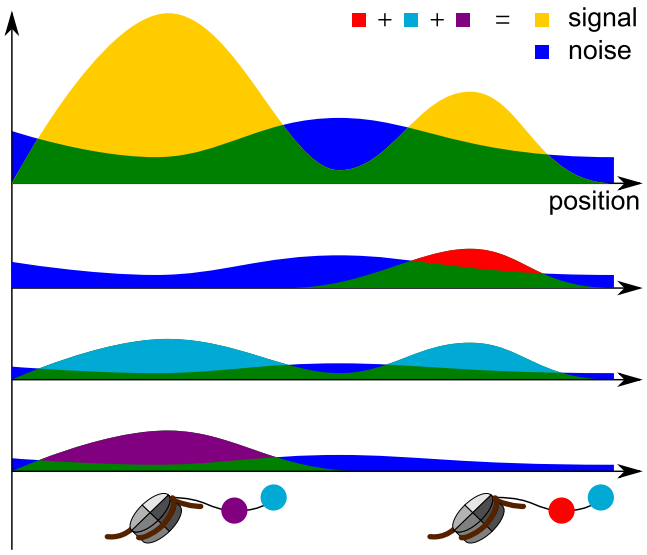

Figure 3.4: *Integration of multiple histone modification experiments. First, peak detection is performed on the sum of the consensus signals, then the signal is decomposed into the contributions of the single histone modifications and a statistical test is performed for each of them to asses whether their contribution is significant or not.*

3.2.6 Inferring the average fragment length

The average fragment length f is typically inferred based on the strand cross-correlation function, defined as:

$$\chi_d = \sum_{i=1}^{w_{\text{gen}}-d} c_{+i} c_{-(i+d)},$$

where d is a positive shift. For point-source factors and low noise levels the cross-correlation function usually has a peak at position f, called the **fragment peak**, as shown in Figure 3.5 (left), which yields a straightforward method for the estimation of f. However for many histone marks the cross-correlation plot is harder to interpret due to the presence of a so-called **phantom peak** [71] and other systematic biases, which can sometimes completely obscure the fragment peak (see Figure 3.5 (right)).

To account for these biases, we introduce a modified cross-correlation function that we call peak cross-correlation:

$$\hat{\chi}_d = \sum_{i=1}^{w_{\text{gen}}-d} \hat{c}_{+i} \hat{c}_{-(i+d)}.$$

The signal $\hat{\boldsymbol{c}}$ is a denoised version of signal $\boldsymbol{c}$ where the read counts are replaced with strand-specific peaks. More specifically, the peak detection technique presented in Section 3.2.2, where it was applied to the consensus signal $\boldsymbol{d}$, here is applied to the raw read counts $\boldsymbol{c}$, considering positive and negative indices independently. The

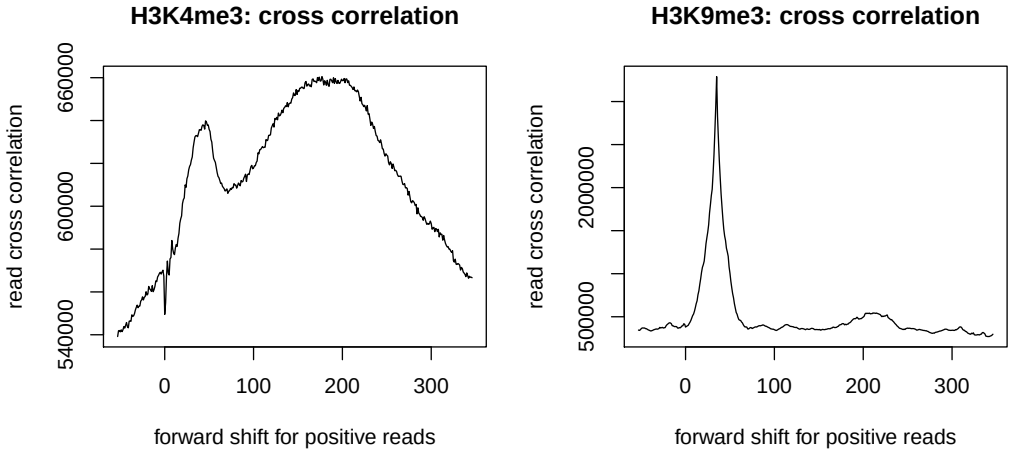

Figure 3.5: *Strand cross-correlation for two different ChIP-seq experiments. Both experiments were done in human K562 cells. On the left, the cross-correlation for the histone modification H3K4me3, which shows both point-source and broad-source peaks. The phantom peak and the fragment peak are clear. On the right, the cross-correlation for the histone modification H3K9me3, with broad-source behaviour. The fragment peak is almost not visible, in contrast with the phantom peak.*

resulting peaks are encoded in the binary vector $\hat{\boldsymbol{c}}$, whose only non-zero entries correspond to peaks' locations.

After the peak cross-correlation function is computed, a clustering technique is applied to interpret it, which yields an estimate for f. Let us consider the peak cross-correlation function $\hat{\boldsymbol{\chi}}$ within an interval $[d_{\min}, d_{\max}]$ (by default from 0 to 300). Let n denote $\sum_{d=d_{\min}}^{d_{\max}} \hat{\chi}_d$. The clustering procedure makes the following assumptions:

1. The observed data is generated by n iid random variables X_i, where each X_i takes on integer values within the range $[d_{\min}, d_{\max}]$. The observations are encoded in $\hat{\chi}_d$, which counts how many X_i take on value d.

2. Each X_i is a mixture of three random variables G_1, G_2 and U. That is, there is a random variable Z that can take on values 1 2 or 3 with probabilities respectively π_1, π_2 and π_3 and such that:

$$X_i|\{Z=j\} \sim \begin{cases} G_1, & j=1 \\ G_2, & j=2 \\ U, & j=3 \end{cases}.$$

   G_1 and G_2 represent respectively the phantom peak and the fragment peak, while U represents the background noise.

3. The random variables G_j, for $j=1$ and $j=2$, are distributed according to a discretized and truncated gaussian random variable with parameters μ_j and

σ_j, that is:

$$\mathrm{P}\{G_j = g\} = \exp\left\{-\frac{(g-\mu_j)^2}{\sigma_j^2}\right\}\left(\sum_{d=d_{\min}}^{d_{\max}} \exp\left\{-\frac{(d-\mu_j)^2}{\sigma_j^2}\right\}\right)^{-1}$$

4. The random variable U is uniformly distributed in the interval $[d_{\min}, d_{\max}]$.

Note that this is an example of the mixture model framework presented in Subsection 2.2.6. See Figure 3.6 for a graphical interpretation.

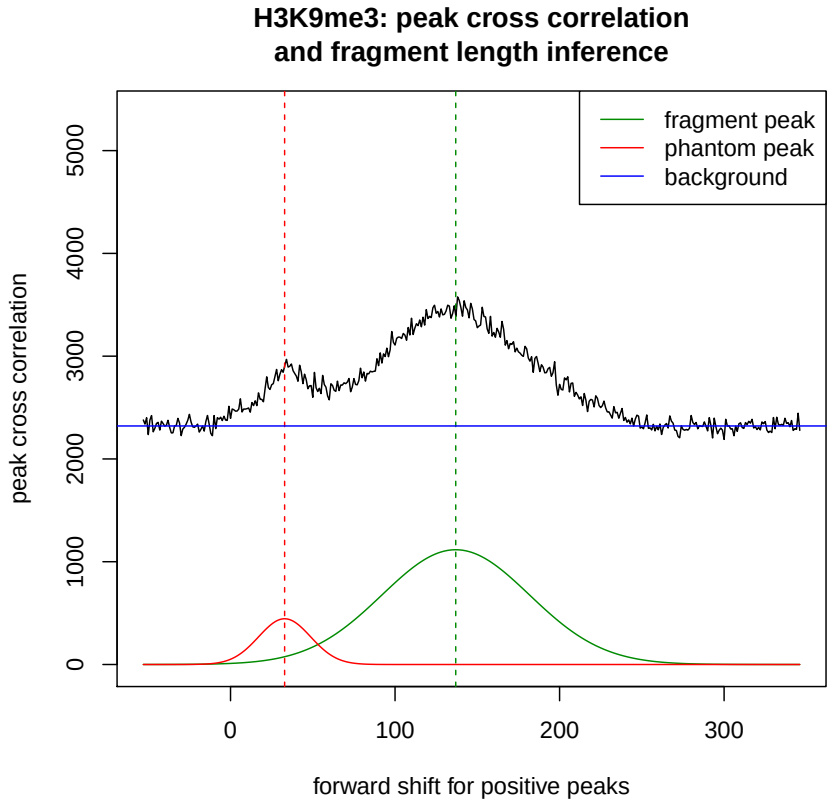

Figure 3.6: *Peak cross-correlation analysis and inference of the fragment length. The sample shown above is the same H3K9me3 sample as in Figure 3.5. The peak cross correlation technique has a strong denoising effect on the cross correlation profile and now also the fragment peak is visible. The colored lines are a graphical representation of the mixture model used for inference. Each colored solid line shows the expected number of samples with a certain forward shift from a random variable in the mixture model: green, red and blue correspond to the random variables G_1, G_2 and U, respectively. The inferred fragment length is shown by the dashed green line.*

To estimate the parameters $\boldsymbol{\pi}$, μ_1, σ_1, μ_2 and σ_2 we use the expectation maximization algorithm presented in Subsection 2.2.7. However, to show how to the

given update rules are used, we need to introduce a slight change of notation. Let $\boldsymbol{x}$ denote an observation vector of length n containing each value d in $[d_{\min}, d_{\max}]$ exactly $\hat{\chi}_d$ times. Then the update rule from Equation 2.8 for the random variables G_1 and G_2 takes the form:

$$\mu_j^{(\text{opt})}, \sigma_j^{(\text{opt})} = \arg\max_{\mu,\sigma} \sum_{i=1}^{n} \gamma_{ij}^{(t)} \log \mathrm{P}\left\{G_j = x_i; \mu, \sigma\right\}.$$

In practice, however, it is unnecessary and inefficient to use a vector of length n, as $\boldsymbol{x}$ is implicitly defined by $\hat{\boldsymbol{\chi}}$. If we denote by i_d an index such that $x_{i_d} = d$ and by $\delta_{dj}^{(t)}$ the quantity $\gamma_{i_d j}^{(t)} \hat{\chi}_d$, the update equation can be rewritten as:

$$\mu_j^{(\text{opt})}, \sigma_j^{(\text{opt})} = \arg\max_{\mu,\sigma} \sum_{d=d_{\min}}^{d_{\max}} \delta_{dj}^{(t)} \log \mathrm{P}\left\{G_j = d; \mu, \sigma\right\}.$$

This optimization problem is solved with a simple gradient descent algorithm. A similar transformation is used also in the update rule for the mixing coefficient.

Since G_1 models the phantom peak commonly observed in cross-correlation analyses, the initial value for μ_1 is set to the average read length, while the initial value for μ_2 defaults to 147 and can be modified by the user. The initial values for σ_1^2, σ_2^2, π_1 and π_2 are respectively $36, 1000, 0.1$ and 0.1.

The phantom peak is not always present. In case the EM algorithm infers an unreasonable value for it (more than 20 bps apart from the average read length), the whole inference is repeated using only the components G_2 and U.

3.3 Results

3.3.1 Comparison to other available tools

We set out to test the predictive power of NucHunter and to compare it with two available nucleosome prediction tools: NPS from Zhang *et al.* [75] and Template Filter from Weiner *et al.* [77]. Other tools had to be excluded from the comparison either because they were not able to deal with the large amount of data or because the results obtained using default parameters were unsatisfactory. Because neither NPS nor Template Filter allow to provide a control sample, for an objective comparison, also NucHunter was used without a control sample.

We evaluated the accuracy of each tool by comparing a set of nucleosome predictions with a gold standard and using different performance measures. First, as a proof of concept, we produced an artificial nucleosome map and we simulated reads according to it. In this assessment, the nucleosomes are predicted from the simulated reads and the artificial nucleosome map serves as a gold standard. Second, we predicted nucleosomes from a published ChIP-seq dataset in yeast, for which a gold standard is publicly available. Third, we tested how consistent the nucleosome predictions are when using replicate datasets. Finally, we analyzed the runtime of the different tools.

3.3.2 Performance measures

Given a list of n_I predicted peaks and a list of n_J peaks from a gold standard, we define three performance measures:

1. the **precision**,
2. the **sensitivity**,
3. the area under the (normalized) error curve (**AUC**).

Let $\text{dist}(i, j)$ denote the genomic distance between the i-th predicted peak and the j-th benchmark peak and let w_{dist} denote a cutoff distance ($w_{\text{dist}} = 20$ bps in our analyses). The precision is the ratio of predicted peaks i such that there is a high-confidence peak j at distance less then w_{dist}:

$$\text{precision} = |\{i \in [1, n_I] : \exists j \in [1, n_J] : \text{dist}(i, j) \leq w_{\text{dist}}\}| {n_I}^{-1},$$

where the $|*|$ operator denotes the cardinality of a set. Similarly, the sensitivity is the ratio of high-confidence peaks j such that there is a predicted peak i at distance less then w_{dist}:

$$\text{sensitivity} = |\{j \in [1, n_J] : \exists i \in [1, n_I] : \text{dist}(i, j) \leq w_{\text{dist}}\}| {n_J}^{-1}.$$

The first performance measure does not penalize situations where many predicted peaks are close to the same benchmark peak and the second one does not penalize situations where for many closely-spaced benchmark peaks there is only one associated prediction.

In order to measure how precisely the predictions match the gold standard, and in line with other studies [86], we use a score that depends on the distribution of the $\text{dist}(i, j)$ values smaller than $w_{\text{AUC}} = 73$ bps (the "errors"). We define the (normalized) cumulative error curve $\text{cum-err}(d)$ as the cumulative distribution function of the errors smaller than the threshold w_{AUC}:

$$\text{cum-err}(d) = \frac{|\{(i, j) : \text{dist}(i, j) \leq d\}|}{|\{(i, j) : \text{dist}(i, j) \leq w_{\text{AUC}}\}|}.$$

Plotting $\text{cum-err}(d)$ versus d, the cumulative error curve should look almost like a $0 - 1$ step for very precise predictions and like a straight line from the origin to the point $(w_{\text{AUC}}, 1)$ for random predictions (see Figure A.1). Therefore, we define the area under the (normalized) error curve AUC as:

$$\text{AUC} = \frac{1}{w_{\text{AUC}} + 1} \sum_{d=0}^{w_{\text{AUC}}} \text{cum-err}(d).$$

Contrary to the sensitivity and precision, the AUC has the property that the predicted peaks and the high-confidence peaks play a symmetric role, i.e. swapping the predictions with the gold standard the result does not change.

3.3.3 The simulated dataset

As a proof of concept, we artificially generated a ChIP-seq sample and a nucleosome map using the following procedure.

1. We considered a chromosome of the length of chromosome IV in yeast (1531933 bps) and reads of 36 bps.

2. We generated reads due to noise. The number of noise reads at each genomic position was sampled from a Poisson distribution with average 2.

3. We generated a nucleosome map. The nucleosome positions were chosen sampling the inter-nucleosomal distance from a geometric distribution with average 18 and adding a minimum distance of 147 base pair.

4. We generated reads due to nucleosomes. Given a fragment length f of 140 base pairs, and for each nucleosome position p, the 5' ends of the reads on the positive and negative strands were generated according to a gaussian random variable with average respectively $p - f/2$ and $p + f/2$ and with uniformly varying sigmas (from 10 to 50). The sampled positions were rounded to the closest integer. The number of sampled reads per nucleosome was chosen according to a Poisson distribution with lambdas such that the expected number of reads at the peak position uniformly varies from 1 to 3.

In Figure 3.7 we show the performance of the three algorithms on the the simulated dataset. For very stringent score cutoffs Template Filter seems to give the most accurate predictions, but its performance declines rapidly when considering a larger number of top-scoring nucleosomes. For less stringent cutoffs NucHunter seems to be more accurate and to reach much higher sensitivity values. However, the model that we used to simulate reads was the same that we had in mind when designing NucHunter, which makes this assessment a proof of concept rather than an objective comparison.

3.3.4 The yeast dataset

We predicted nucleosomes based on a ChIP-seq experiment for the histone mark H3K9ac from yeast [87] and we validated them with a high resolution map of nucleosome positions[88]. This map has been obtained with a technique that is independent from ChIP-seq, and it is claimed to be more accurate.

The results from Figure 3.8 show that NucHunter makes more accurate predictions compared with the other tools. Considering the default score thresholds, NucHunter and NPS return a similar number of predictions but the former has an higher AUC, whereas Template Filter returns many more predictions, but of lower quality. When the score threshold is increased, the AUC difference between NucHunter and NPS becomes much more pronounced. This suggests that the nucleosome predictions with highest score from NucHunter are, in general, much more

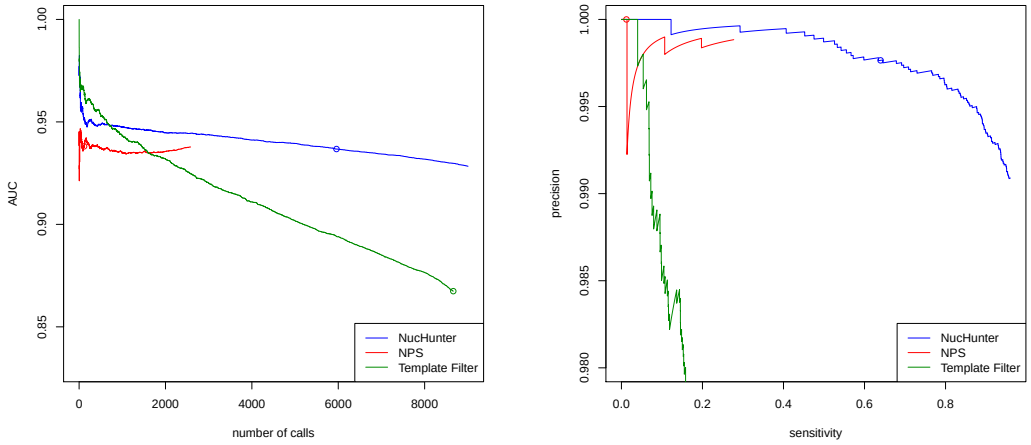

Figure 3.7: *Performance of the three algorithms on the simulated dataset. The performance measures (AUC, sensitivity and precision) are computed for every possible score threshold, which results in a AUC-number of calls curve (left) and a precision-sensitivity curve (right). The circles indicate the performance of the algorithms using the default thresholds.*

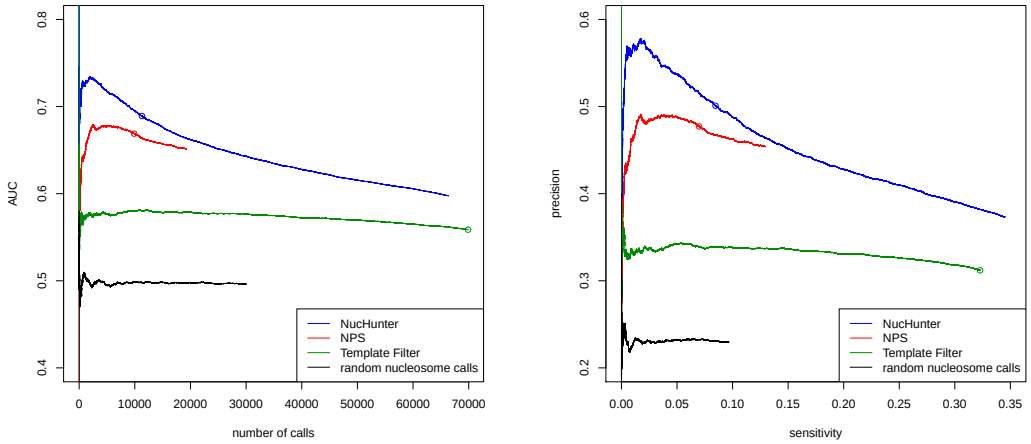

Figure 3.8: *Performance of the three algorithms on the yeast dataset. The performance measures (AUC, sensitivity and precision) are computed for every possible score threshold, which results in a AUC-number of calls curve (left) and a precision-sensitivity curve (right).*

precise compared with those from the other tools. When the default score thresholds are used, all the tools suffer from low sensitivity in this dataset, in particular NucHunter and NPS.

There can be different reasons for unidentified nucleosomes or incorrect predictions. In the first place, the experimental procedures used for the ChIP-seq experiment and that used for the gold standard are different. Roughly 5.6% of the nucleosomes in the gold standard, for instance, are located in low-mappability regions and are not covered by any read. Moreover, the ChIP-seq experiment targeted only acetylated nucleosomes, as opposed to the gold standard. A more general problem is the identification of fuzzily positioned nucleosomes. If the nucleosome positioning varies extensively from cell to cell, the assumptions made by the algorithms are violated and nucleosomes are hard to identify. Lastly, both precision and sensitivity are affected from high noise levels, insufficient sequencing coverage and sequencing biases.

3.3.5 Consistency on replicate datasets

The publicly available epigenomic data for the human leukemia cell line K562 [70] includes replicate ChIP-seq experiments for different histone marks. We used these samples to test how consistent the nucleosome calls are between replicates. As a consistency score, we considered the t nucleosome predictions with highest score from each of the two replicates and we computed an AUC value from these two sets of nucleosomes, for each possible value of t.

Figure 3.9 shows how the AUC between predictions from two ChIP-seq samples depends on the total number of nucleosome calls. There are some histone marks where NucHunter is not the most consistent algorithm, such as H3K27me3, H3K36me3 and H3K9me3. These, however, are all broad marks, i.e. associated to nucleosomes that are not well positioned and that NucHunter is not designed to detect. With the marks H3K4me3, H3K27ac, H3K9ac and H2A.Z, which are known to be associated to well positioned nucleosomes, NucHunter seems to be the most consistent among the three algorithms.

3.3.6 Runtime comparison

Computational efficiency was among the design principles that guided the development of NucHunter. To assess how efficiently NucHunter can detect nucleosomes in large genomes, we compared its runtime with that of NPS and Template Filter. We used a ChIP-seq sample for histone modification H3K4me3 from the K562 dataset mentioned in Subsection 3.3.5 and we ran the tools separately for each chromosome. Splitting the reads by chromosome was necessary in order to measure the runtime of Template Filter, which otherwise would not run. All tests were carried out on a AMD Opteron computer with a clock speed of 2.66 GHz. Overall, the results in Table 3.1 show that NucHunter is considerably faster than the other two algorithms.

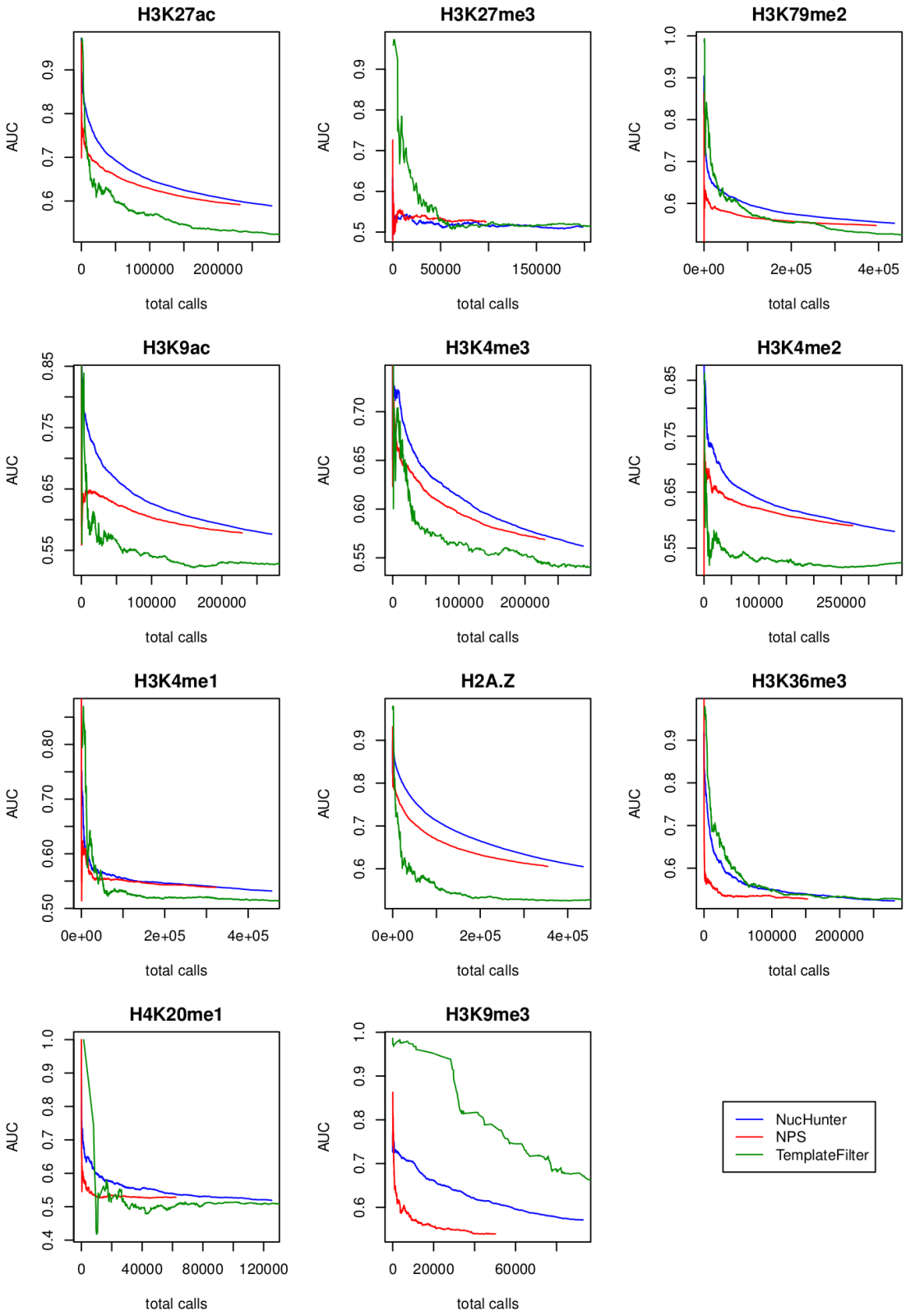

Figure 3.9: *Performance of the three algorithms on replicate experiments. The end points of the curves show the total number of calls and the AUC using the default score thresholds.*

chromosome	size	total reads	NucHunter	NPS	Template Filter
chr1	249250621	2390956	52.50	546.42	21033.22
chr2	243199373	1670941	50.89	376.14	13375.48
chr3	198022430	1349593	38.12	301.39	10169.83
chr4	191154276	1136302	34.96	246.71	7069.05
chr5	180915260	1251884	34.64	278.19	7712.84
chr6	171115067	1661990	34.65	384.34	11558.63
chr7	159138663	1392134	37.66	374.89	7020.56
chr8	146364022	948543	34.90	205.60	3990.71
chr9	141213431	720819	66.57	147.95	2285.28
chr10	135534747	929571	34.95	204.00	3432.10
chr11	135006516	1066208	32.32	219.01	3881.10
chr12	133851895	1037149	27.61	213.28	3592.43
chr13	115169878	451543	26.03	103.08	1561.54
chr14	107349540	486553	23.40	90.36	1407.95
chr15	102531392	645344	32.77	139.30	1730.24
chr16	90354753	677344	29.75	130.39	1573.29
chr17	81195210	797666	25.84	148.33	1638.95
chr18	78077248	448191	22.54	102.60	1219.75
chr19	59128983	770912	19.79	122.11	1068.30
chr20	63025520	443351	19.90	85.54	895.27
chr21	48129895	306180	19.22	75.84	568.34
chr22	51304566	386873	21.36	77.54	548.08
chrX	155270560	593859	41.63	106.96	2311.36

Table 3.1: *Runtime comparison on one-chromosome files derived from a H3K4me3 ChIP-seq experiment in human K562 cells. The columns specify respectively the chromosome, the chromosome size in base pair, the number of reads mapped to the chromosome, and the running time of the three algorithms in seconds.*

3.3.7 Clustering of nucleosomes based on histone marks

We ran NucHunter on the K562 dataset mentioned in Subsection 3.3.5 consisting of a control experiment and 12 ChIP-seq experiments for distinct histone modifications. Differently than in the consistency assessment, we considered only the first replicate for each mark and we used all marks simultaneously. NucHunter returns, along with other statistics, the raw read count within a window of a specified radius w_{enr} around the inferred nucleosome location, for each detected nucleosome and for each experiment. This data can be conveniently represented as a matrix with as many rows as the detected nucleosomes and as many columns as the number of histone marks (the count matrix). Additionally, when a control experiment is present, NucHunter also returns the noise level, which is computed from the control experiment for each detected nucleosome. The noise level can be conveniently represented as a vector with as many elements as the detected nucleosomes. We used these data for an exploratory analysis of the chromatin landscape.

In order to make the read counts a suitable input for the k-means clustering algorithm, we devised an ad hoc normalization procedure for the count matrix that takes into account the noise levels, the different sequencing depths of the datasets and the nucleosome abundance at each locus (see Mammana *et al.* [1] for the details). The normalized count matrix is finally used as input for the k-means clustering algorithm.

Given a parameter n_{clust}, this unsupervised learning method aims at partitioning the data points into n_{clust} different families (clusters) such that elements in the same cluster are as similar to each other as possible. Each cluster is characterized by its centroid, which is, in our case, a prototypical histone modification pattern.

We found that with n_{clust} equals 6 the results are robust, whereas for higher values of the parameter the clusters tend to change depending on the initialization. Moreover and most importantly we found that such a partitioning, derived solely from the histone modification patterns, can also capture biologically meaningful positional features of the nucleosomes. We assigned labels to each cluster based on the histone modification pattern and genomic localization. The labeled centroids are shown in Figure 3.10.

We studied the genomic localization of nucleosomes from the different clusters using the RefSeq annotation dataset as well as publicly available data from cap analysis of gene expression (CAGE) and DNase I hypersensitivity sequencing experiments [70]. More precisely, we computed the following statistics: (i) we derived a consensus nucleosome profile along genes by considering a large set of annotated genes, by rescaling their nucleosome profiles to the same length and by adding them up (Figure 3.11); (ii) we analyzed the nucleosome positioning around promoters of active genes by considering the distribution of distances between CAGE tags and nucleosomes (Figure 3.12 (left)); (iii) we obtained the average DNase I hypersensitivity profile around nucleosomes for each class (Figure 3.12 (right)).

Overall these statistics give a clear picture of the nucleosome landscape and recapitulates previous knowledge (see Figure 3.10). The nucleosomes in the first family are characterized by a strong enrichment of H3K4me2/3 and H3K9ac and they tend to reside in the 5' portion of a gene near the transcription start site (TSS; Figure 3.11). Thus, we labeled them **promoter** nucleosomes. In proximity of promoters of active genes, these nucleosomes exhibit a strikingly regular pattern (Figure 3.12 (left)), whose main features are a nucleosome-depleted region right upstream the TSS and a well positioned nucleosome 170 bps downstream (the +1 nucleosome). The second and third clusters show an enrichment of H3K4me1 and H2AZ as well as a general enrichment of active marks, whereas TSS-associated histone marks, such as H3K4me2/3 and H3K9ac, are less enriched compared with the promoter cluster. These features, together with the high levels of DNase I hypersensitivity that we observe (Figure 3.12 (right)), suggest that these nucleosomes may flank enhancer sequences. Thus, we labeled them as **enhancer 1** and **enhancer 2** nucleosomes. The fourth centroid is enriched in H3K79me2 and H4K20me1, whereas the fifth centroid is enriched in H3K36me3 and H3K9me1, which are all histone marks related to elongation of RNA polymerase II [89]. The localization of these two classes of elongation nucleosomes along the gene body, shown in Figure 3.11, suggests that

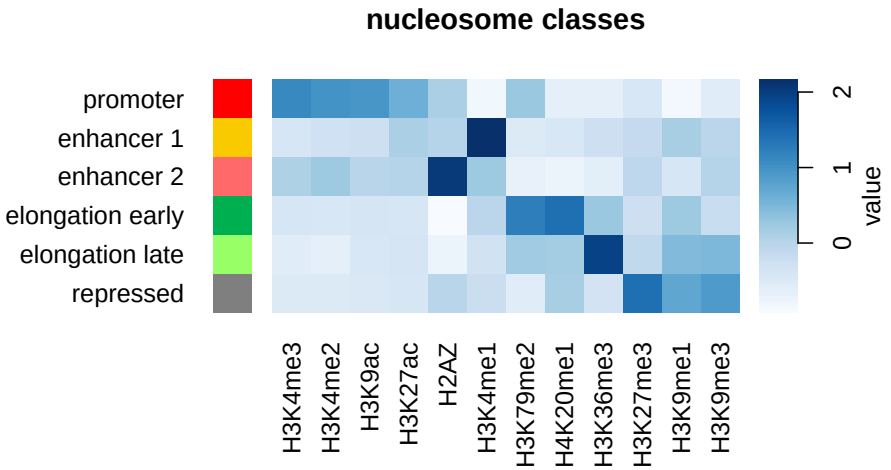

Figure 3.10: *Centroids obtained from the k-means clustering algorithm. A total of 422547 nucleosomes called by NucHunter was clustered into 6 clusters: promoter (20.4%), enhancer 1 (19.8%), enhancer 2 (14.4%), elongation early (16.4%), elongation late (14.7%) and repressed (14.3%). The rows of the heatmap represent the centroids of the clusters and the columns represent the histone modifications. The labels have been assigned based on prior biological knowledge.*

the 5th centroid is enriched toward the 3' end of a gene, whereas the 4th centroid is enriched more to the 5' end. Thus, we termed them **elongation early** and **elongation late** nucleosomes, respectively. The last centroid is characterized by an enrichment of H3K9me3 and H3K27me3, suggesting that it represents chromatin-repressed genomic regions [90]. Thus, we termed it **repressed**.

3.4 Discussion

NucHunter is an algorithm that detects a specific pattern in ChIP-seq experiments for histone modifications, where an occurrence of the pattern is evidence for the presence of a positioned nucleosome. A more accurate and efficient signal processing and an improved statistical analysis of the peaks count among the strengths and the innovative aspects of this tool.

Another innovative aspect is NucHunter's capability of integrating several ChIP-seq experiments at once. ChIP-seq experiments for histone marks, in fact, provide data only for those nucleosomes where the targeted mark is present. Therefore, combining samples targeting different histone marks is essential for deriving a comprehensive nucleosome map. Most importantly, such a combined input does not only detect where nucleosomes are positioned, but also how they are modified. This information can be used to distinguish different classes of nucleosomes and relate each class to a genomic context.

NucHunter, however, is only a first step in the exploration of the patterns contained in the count signals and it opens up new important challenges. The clustering

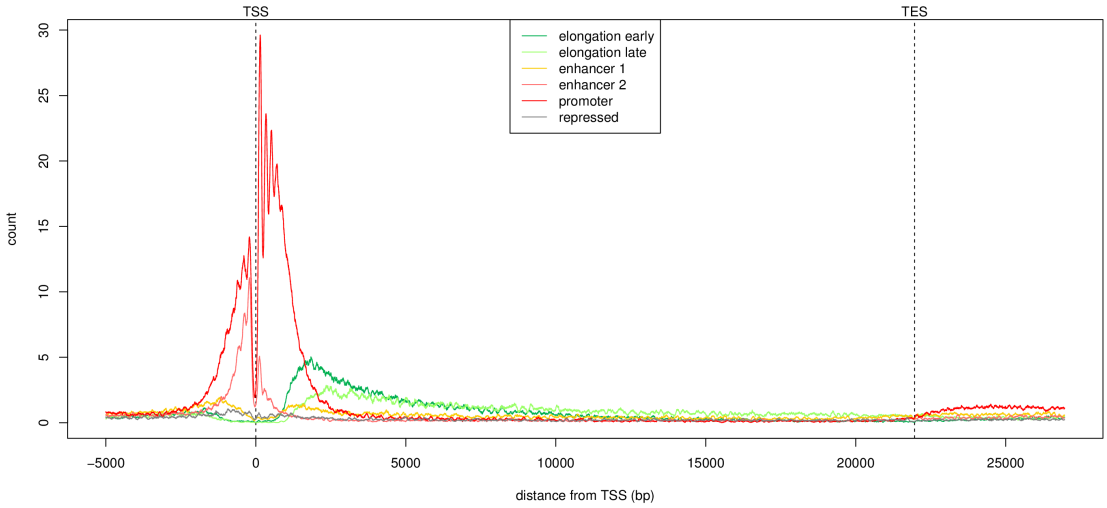

Figure 3.11: *Nucleosome occupancy along the gene body. The nucleosome occupancy profiles from a subset of genes in RefSeq have been rescaled to the same length and summed up. For each class a separate profile has been computed.*

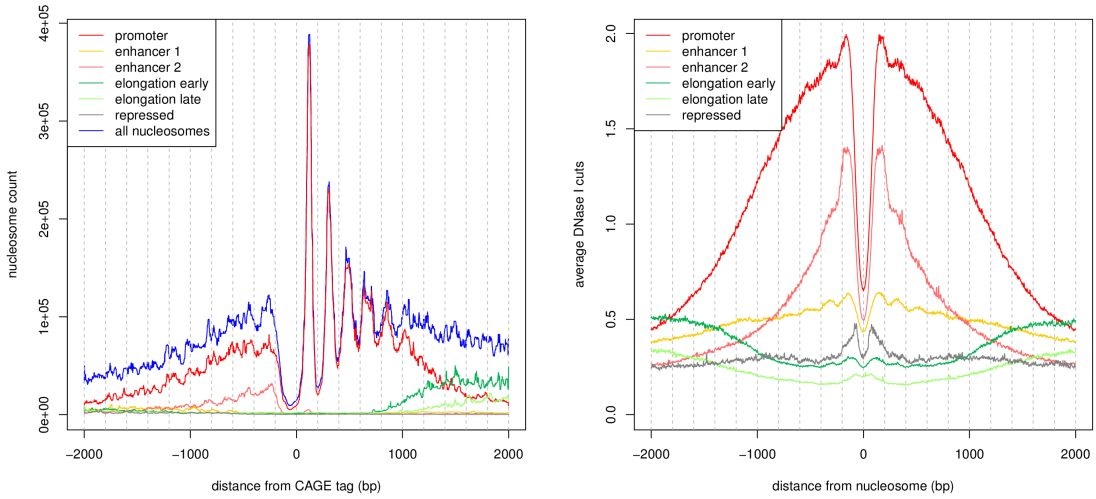

Figure 3.12: *CAGE and DNase I hypersensitivity reads around nucleosomes. On the left, nucleosome distribution at promoters of active genes. The profile has been obtained by computing the distribution of distances between CAGE tags and nucleosomes. On the right, DNase I hypersensitivity levels in relation to nucleosomes. The profile for each nucleosome class is the average DNase I hypersensitivity profile of all nucleosomes from that class.*

analysis shown in the Results section, for instance, showed that multiple ChIP-seq experiments can be used to identify nucleosome classes associated to important biological processes. This classification, however, was ad hoc, in that it required a number of arbitrary choices regarding the data processing and the clustering method. Moreover, in regions where nucleosomes are not well positioned or the read counts are too low for the precise nucleosome position to be determined, no information about the chromatin landscape can be derived. This motivates the development of a more principled approach for studying the state of the chromatin which can be used to annotate the whole genome. This topic is the subject of Chapter 4.

Another important open problem is the choice of the optimal filter for the feature under consideration. The Mexican hat wavelet was chosen mostly based on expert knownledge, with the average fragment length for each experiment being the only parameters fitted to the data. A more data-driven approach is difficult mostly because of two limitations intrinsic to the nucleosome detection problem from ChIP-seq data. The first one is that the feature to be detected cannot be accurately defined, as the separation between point-like and highly dynamic and heterogeneous nucleosome positioning is not clear. The second one is related to the ChIP-seq protocol. In fact, even if nucleosomes would always form at the same precise positions in a cell population, the peaks in the count signals would be too broad and would not contain enough information for the original positions to be precisely inferred. Chapter 5 explores another pattern recognition problem where the feature of interest can be more accurately defined and the sequencing protocol has a higher resolution compared to ChIP-seq.

Chapter 4

Chromatin segmentation

This chapter presents EpiCSeg: an algorithm that integrates several ChIP-seq experiments for histone marks and characterizes a given number of chromatin states automatically. Because these states are associated to specific biological processes, they can be used to annotate the genome.

4.1 Motivation

A central question in biology is how cells of a multicellular organism with essentially the same genotype can establish and maintain distinct phenotypes. Because the genome cannot be associated to this variability, current research is focused on the epigenome.

Today many consortia, such as NIH Roadmap Epigenomics, ENCODE, Blueprint, DEEP and IHEC [69, 70, 91–93], are providing genome-wide maps of histone modification generated using the ChIP-seq technique. Typically, for a given cell type, a panel of histone modifications are profiled in order to gain insight into the cell-type specific epigenome. This huge amount of available data calls for the development of integrative computational approaches to identify the most important, biologically meaningful features and to capture recurrent patterns.

The segmentation of epigenomes into chromatin states aggregates the ChIP-seq tracks and provides an abstract view on the multidimensional data. A chromatin state is a recurrent pattern in the abundances of a given set of histone modifications, typically related to a particular biological function. Chromatin segmentation aims at explaining the observed epigenomic data as a sequence of hidden chromatin states, where the number of distinct possible states is small. The idea of chromatin segmentation is not new [94–98], however, the small number of available computational tools for this task and the growing importance of epigenomic datasets suggest that there are still ample margins for improvement. Two popular tools are ChromHMM [95] and Segway [97]. In both approaches, the ChIP-seq experiments are transformed into genome-wide multivariate signals and subsequently used as observed variables in a probabilistic inference algorithm.

In ChromHMM the raw reads are assigned to non-overlapping bins of 200 bps

and a sample-specific threshold is used to transform the count data to binary values. Given a hidden state, the binary vectors are modeled as independent Bernoulli random variables. This approach has some limitations. (i) There is a considerable loss of information when transforming a read count into a binary value, as the possibility of distinguishing between different levels of activity is precluded. This limitation is especially important for more recent, higher coverage ChIP-seq experiments. (ii) There is no obvious way of deciding which threshold to use, despite it being critical for the final segmentation. (iii) The independent Bernoulli model assumes independence between the chromatin marks given a hidden state. That would imply, for instance, that in those regions where a promoter state occurs, the presence of the mark H3K4me3 is independent from the presence of the mark H3K27ac, which is in contrast to our observations (see Subsection 4.2.2). (iv) A large portion of the genome is assigned to a state with no clear role, apart from being associated to read counts below the discretization threshold.

Segway works at a single base-pair resolution and transforms the counts into real values. Given a hidden state, a vector of transformed read counts is modeled with independent gaussian random variables. The following shortcomings can be noted: (i) as in ChromHMM, the independence assumption between marks seems inadequate, (ii) the choice of the monotone function is not easy to justify, especially because the resulting zero-inflated distribution can be very different from a gaussian distribution, (iii) because it works at a single base pair level, this method is orders of magnitude slower than ChromHMM, which severely limits its applicability.

In order to address these shortcomings we developed Epigenome Count-based Segmentation (EpiCSeg), a segmentation algorithm with the following main features. (i) Raw read counts can be directly used as observation symbols, thus eliminating the need for preprocessing steps and arbitrary thresholds. (ii) An accurate discrete multivariate probability distribution is used for modeling the count vectors given a hidden state, which can recapitulate the overdispersion and correlation features observed in the data. (iii) The probabilistic framework and computational efficiency are similar to those of ChromHMM, making EpiCSeg useful also for large genomes, such as the human genome.

4.2 Methods

EpiCSeg's main feature is the multivariate modeling of read counts from several histone marks, which is then integrated in a hidden Markov model (HMM) to produce a segmentation of the genome. The input of the algorithm is a desired number of states and a count matrix where each element is the number of reads of a certain mark in a certain genomic bin. The output of the algorithm is, among other things, a vector that assigns each genomic bin to one of the states. EpiCSeg automatically learns states that are associated to important biological processes (see Figure 4.1).

In this section we will first briefly describe how the count matrix is obtained from the reads, then we will present EpiCSeg's probabilistic model in detail, and finally we will conclude the section with some computational considerations.

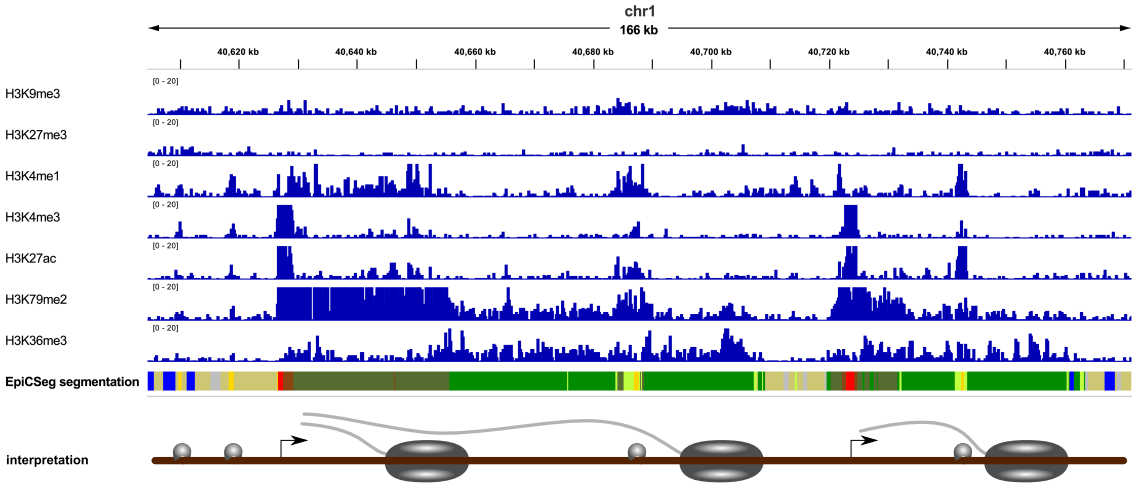

Figure 4.1: *EpiCSeg simplifies the interpretation and the analysis of ChIP-seq data. The first 7 tracks represent the input data, that is, read counts for each genomic bin and for a panel of histone marks. The 8th track represents the main output of the algorithm, that is, a partition of the genome into segments of different types, where each type is a chromatin state and it is represented by a specific color in this picture. The drawing at the bottom illustrates how the states identified by EpiCSeg can be interpreted biologically. The red state can be interpreted as the beginning of a gene (drawn as an arrow), the green states can be interpreted as transcribed regions (represented by an RNA polymerase II producing a transcript) and the yellow regions can be interpreted as enhancers (drawn as a transcription factor binding to the DNA).*

4.2.1 From reads to read counts

A unique feature of EpiCSeg is that the input data can be derived from the mapped reads directly, almost without any preprocessing. The genomic regions of interest, which can be whole chromosomes, or better yet, only assembled and mappable regions, are partitioned into non-overlapping subregions of the same size called bins, and then each read is assigned to one bin. Typically, bins have a size of 200 base pairs, though this is an adjustable parameter. Using counts per bin instead of counts per base pair not only has the practical advantage of reducing the runtime significantly, but it also smooths the input data conveniently.

To decide to which base pair, and therefore to which bin, a read belongs to, a very similar approach as in NucHunter is used (see Section 3.2.1). Briefly, for single-end ChIP-seq libraries, each read is assigned to its 5' coordinate shifted in the 5'-to-3' direction by about half of the estimated average fragment length. This coordinate is, in fact, an estimate of the midpoint of the DNA fragment that the read originates from. The appropriate shift can be estimated using NucHunter (as we do in the Results section), or by using the default value of 75 base pairs, which will be appropriate in most cases. For paired-end ChIP-seq libraries, EpiCSeg counts directly the fragments' midpoints. In both cases, read counting is done using the Bioconductor package bamsignals [3].

4.2.2 A multivariate probabilistic model for read counts

The identification of chromatin states is a complex and ill-defined problem. EpiCSeg's approach is to define a probabilistic model for the observed counts for each chromatin state and to integrate these in a hidden Markov model (HMM, presented in Subsection 2.2.8). Following this strategy the central problem becomes: "given a chromatin state, what read counts do we expect to observe?".

This is a considerable challenge for at least two reasons. First, because the distribution of the read counts from replicate experiments in the same region is **overdispersed**, i.e. its variance is so large that a simple Poisson model cannot account for it [52]. This degree of variation is also important when modeling the read counts associated to the same chromatin state. Second, because the read abundances along the genome tend to be **correlated**. This can happen because of technical or biological biases, such as mappability, chromatin accessibility and unspecific antibody binding [82], but it can also be a reflection of the biological processes taking place on the chromatin fiber. The histone modification abundances at promoters, for instance, have been shown to accurately predict the expression levels of genes [99]. Therefore it should be expected that in a given chromatin state the mark abundances vary more or less in proportion to the activity of the biological process they are related to.

It is because of the correlation and overdispersion properties, as illustrated in Figure 4.2, that EpiCSeg uses a negative multinomial distribution to model the reads observed in each hidden state (see Subsection 2.2.3). In the next subsection we will formalize EpiCSeg's complete probabilistic model.

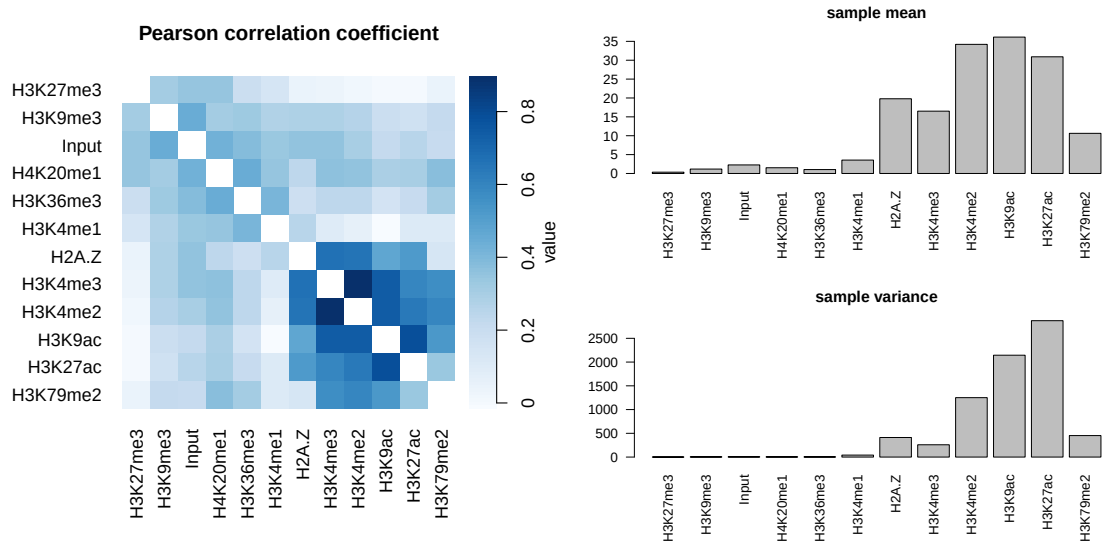

Figure 4.2: *Correlation and overdispersion in the components of a chromatin state. The count data was derived from the K562_1 dataset presented in Section 4.3. Only bins annotated as DNase+TSS were considered, corresponding to promoters of actively transcribed regions (see Subsection 4.3.1). This allows to have an idea of how the counts are distributed given a chromatin state, without using EpiCSeg for defining chromatin states. The plot on the left shows the Pearson correlation coefficient between every pair of histone marks (the coefficients on the diagonal, which equal 1 by definition, have been set to 0 for display purposes). It can be noted that most of the correlations are considerably higher than 0, in agreement with the negative multinomial model. The plot on the right shows the mean and the variance of the counts per bin. The variance is considerably higher than the mean, indicating that a Poisson model is inadequate and that the data is overdispersed.*

4.2.3 The hidden Markov model

EpiCSeg is based on the hidden Markov model framework presented in Subsection 2.2.8. Here we briefly describe and explain the parameters of the model, including the emission parameters. We will denote by n the total number of bins and by m the number of histone marks analyzed. For ease of discussion we will assume that only one genomic region is considered. The input data is represented by a count matrix $\boldsymbol{c}$ with n rows and m columns. The i-th row of the count matrix, where $1 \leq i \leq n$, will be denoted as $\boldsymbol{c}_i$, the count in the i-th bin corresponding to the l-th mark, where $1 \leq l \leq m$, will be denoted as c_{il} and the sum of the counts in the i-th row will be denoted as c_{i+}, i.e. $c_{i+} = \sum_{l=1}^{m} c_{il}$.

The main assumption of the model is that each of the n observations $\mathbf{c}_i$ is explained by a corresponding hidden state. There are k possible states, each of them represents a chromatin state and it is modeled by a negative multinomial distribution. The random variable Z_i denotes the unknown, hidden state at position i and the random variable X_i represents the observation at position i, which takes the value $\boldsymbol{c}_i$.

The complete set of model parameters, denoted as θ, consists of the parameters $\boldsymbol{\pi}, \boldsymbol{\alpha}, \nu_1, \nu_2, ...\nu_k$, where:

1. $\boldsymbol{\pi}$, the initial probabilities, are a vector of k probabilities summing up to one where π_j specifies $\mathrm{P}\{Z_1 = j\}$.

2. $\boldsymbol{a}$, the transition probabilities, are a square matrix of size k, where the element in row u and column v, denoted as a_{uv}, specifies $\mathrm{P}\{Z_{i+1} = v | Z_i = u\}$, independently of the position i in the sequence. In $\boldsymbol{a}$ each row u sums up to one, i.e. $\sum_{v=1}^{k} a_{uv} = 1$.

3. ν_j is the parameter set that determines the emission probabilities relative to state j, that is, the parameters $\mu_j, r_j, p_{j1}, p_{j2}, ...p_{jm}$ characterizing a negative multinomial distribution (see Subsection 2.2.3). In formulas:

$$\mathrm{P}\{X_i = \boldsymbol{c}_i | Z_i = j\} = f_{\mathrm{NM}}(\mathbf{c}_i; \nu_j) = f_{\mathrm{NB}}(c_{i+}; \mu_j, r_j) f_{\mathrm{Multinom}}(\mathbf{c}_i; p_{j1}, p_{j2}, ...p_{jm}), \tag{4.1}$$

where

$$f_{\mathrm{NB}}(c_{i+}; \mu_j, r_j) = \frac{\Gamma(r_j + c_{i+})}{\Gamma(r_j) c_{i+}!} \left(\frac{\mu_j}{\mu_j + r_j}\right)^{c_{i+}} \left(\frac{r_j}{\mu_j + r_j}\right)^{r_j},$$

and

$$f_{\mathrm{Multinom}}(\mathbf{c}_i; p_{j1}, p_{j2}, ...p_{jm}) = c_{i+}! \prod_{l=1}^{m} \frac{p_{jl}^{c_{il}}}{c_{il}!}.$$

In an important variant of the EpiCSeg model all r_j variables are constrained to have the same value. This strategy has proved effective in avoiding overfitting and excluding some unrealistic models where different states j have wildly different values of the parameter r_j. This variant of the model is called the **dependent mode**, while the variant where the r_j parameters are independent is called the

independent mode (see Figure 4.3). Unless otherwise specified this discussion focuses on the independent mode.

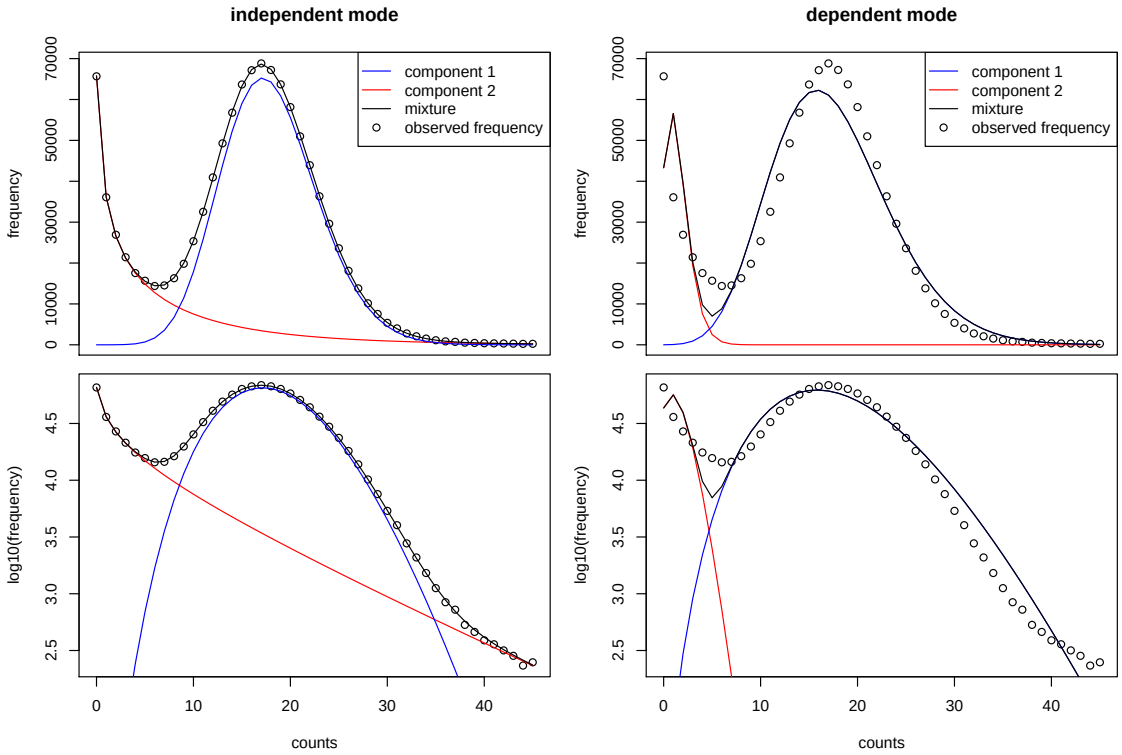

Figure 4.3: *EpiCSeg's dependent and independent mode. In this example a large number of counts has been generated according to a bimodal distribution, whose frequencies are shown by the empty circles. Fitting a mixture model with two negative binomial distributions, intuitively, should separate the two modi of the distribution. However, if the dispersion parameters of the two distributions are unlinked and completely free to be fitted (the independent mode, shown on the left), counterintuitive results can be obtained. On the left, the component colored in red accounts for the first modus, as well as for the tail of the second modus, while the component colored in blue accounts for most of the second modus. This is generally undesirable when performing clustering. On the right, the two negative binomials have been fitted by imposing that the dispersion parameters are the same (dependent mode). Even though the overall fit and the likelihood of the model is not as good as for the independent mode (compare the black solid line with the black circles), now each modus can be associated to a distinct component of the mixture model and the results can be used for clustering. This example deals with univariate observations and mixture models, rather than multiple histone marks and hidden Markov models, but the same ideas apply to both cases (which are both handled by EpiCSeg).*

4.2.4 Update rules for the negative multinomial distribution

EpiCSeg learns all parameters using unsupervised learning and the Baum-Welch algorithm presented in Subsection 2.2.9. At each iteration t, therefore, the forward-backward algorithm is used to compute the posterior probabilities $\gamma_{ij}^{(t)}$, defined as:

$$\gamma_{ij}^{(t)} = \mathrm{P}\left\{Z_i = j | \boldsymbol{X} = \boldsymbol{c}; \theta^{(t)}\right\},$$

where $\theta^{(t)}$ are the current parameter estimates and $\boldsymbol{X} = (X_1, X_2, ...X_n)$. We already showed in Subsection 2.2.9 how the initial probabilities and the transition probabilities are updated. In this subsection we show how the emission parameters are updated, or, equivalently, how the optimization problem from Equation 2.10 is solved:

$$\nu_j^{(t+1)} = \arg\max_{\nu_j} \sum_{i=1}^{n} \gamma_{ij}^{(t)} \log f_{\mathrm{NM}}(\mathbf{c}_i; \nu_j).$$

As discussed in Subsection 2.2.9, the solution to this problem can also be used for mixture models, which have also been implemented in EpiCSeg.

Because of the factorization shown in Equation 4.1, the parameters of the negative binomial and those of the multinomial distributions can be maximized independently:

$$\boldsymbol{p}_j^{(t+1)} = \arg\max_{\boldsymbol{p}_j} \sum_{i=1}^{n} \gamma_{ij}^{(t)} \log f_{\mathrm{Multinom}}(\boldsymbol{c}_i; \boldsymbol{p}_j),$$
$$\mu_j^{(t+1)}, r_j^{(t+1)} = \arg\max_{\mu_j, r_j} \sum_{i=1}^{n} \gamma_{ij}^{(t)} \log f_{\mathrm{NB}}(c_{i+}; \mu_j, r_j).$$

The following closed-form solution hold for the parameters $\boldsymbol{p}_j$ of the multinomial distribution :

$$p_{jl}^{(t+1)} = \frac{\sum_{i=1}^{n} \gamma_{ij}^{(t)} c_{il}}{\sum_{i=1}^{n} \gamma_{ij}^{(t)} c_{i+}}.$$

In fact, the optimization function can be written as:

$$\sum_{i=1}^{n} \gamma_{ij}^{(t)} \log f_{\mathrm{Multinom}}(\boldsymbol{c}_i; \boldsymbol{p}_j) = \sum_{i=1}^{n} \gamma_{ij}^{(t)} \log \frac{c_{i+}!}{\prod_{i=1}^{m} c_{il}!} + \sum_{l=1}^{m} \left(\sum_{i=1}^{n} \gamma_{ij}^{(t)} c_{il} \right) \log p_{jl}.$$

In the last expression, the first term is independent of $\boldsymbol{p}_j$ while the second can be maximized applying Lemma B.1.2.

For the parameter μ_j of the negative binomial distribution, the following closed-form solution can be proven using standard analytical techniques (see Lemma B.1.3):

$$\mu_j^{(t+1)} = \frac{\sum_{i=1}^{n} \gamma_{ij}^{(t)} c_{i+}}{\sum_{i=1}^{n} \gamma_{ij}^{(t)}}.$$

The the optimal r_j parameters are given by:

$$r_j^{(t+1)} = \arg\max_{r_j} \sum_{i=1}^{n} \gamma_{ij}^{(t)} \log f_{\mathrm{NB}}(c_{i+}; \mu_j^{(t+1)}, r_j). \tag{4.2}$$

Unfortunately there is no closed formula for the last maximization problem, which needs to be solved numerically as a one dimensional optimization problem. It is known, however, that there exist only one local maximum, which is therefore a global maximum [100]. In EpiCSeg a variant of the Brent algorithm is used for this task [101].

In the dependent mode of the EpiCSeg model, i.e. when all parameters r_j are constrained to have the same value r, all the above update equations remain valid except Equation 4.2, which becomes

$$r^{(t+1)} = \arg\max_{r} \sum_{j=1}^{k} \sum_{i=1}^{n} \gamma_{ij}^{(t)} \log f_{\mathrm{NB}}(c_{i+}; \mu_j^{(t+1)}, r), \tag{4.3}$$

and can be solved numerically as in the previous case.

4.2.5 The initialization algorithm

The Baum-Welch algorithm used by EpiCSeg for fitting the hidden Markov model needs an initial value for all the model parameters. Those parameters will eventually be fit to the data, but the Baum-Welch algorithm can converge to different local maxima depending on the initialization. The initialization procedure aims at finding initial parameters such that the final ones are close to the global maximum point. In EpiCSeg, this procedure is based on the following ideas:

- The parameters for k chromatin states can be initialized by clustering the observation vectors into k clusters and therefore disregarding the order of the observations.

- Many small clusters (seeds) can be merged and reduced to k clusters using hierarchical clustering.

- Principal component analysis (PCA) can be applied on the count matrix to find good seeds.

Let k be the desired number of clusters and $\boldsymbol{c}$ the count matrix with m columns and n rows. $\boldsymbol{c}$ and k are the input of the algorithm while the output is a different set of bins for each cluster. These sets do not constitute a partitioning of all bins, because they can overlap, and they are not even proper sets, but rather multisets, as the same bin can be counted multiple times. From each multiset the emission parameters for a state can be easily fitted using maximum likelihood.

The algorithm performs the following steps:

1. PCA is performed on the count matrix $\boldsymbol{c}$ (see Figure 4.4). This results in a coordinate matrix $\dot{\boldsymbol{c}}$ with the same dimensions as $\boldsymbol{c}$ but where the columns correspond to principal components (PCs) .

2. For each PC, n_{lev} seeds are computed (see Figure 4.5). Here seed means a subset of the bins and n_{lev} is a parameter of the initialization algorithm. The seeds deriving from a PC p result from partitioning the rows of matrix $\dot{\boldsymbol{c}}$ into n_{lev} groups of equal size according to the intensity of the p-th column. So, for instance, the first group is formed by the n/n_{lev} rows with the lowest values in column p. This yields a total of $n_{\text{lev}} \cdot k$ seeds.

3. For each seed the parameters of a negative multinomial distribution $\text{NM}(\mu, r, \boldsymbol{p})$ are determined. The r parameter is shared among all seeds and fitted by assuming that all bins of the count matrix are generated by the same negative multinomial distribution. The other parameters are estimated by maximum likelihood on the bins identified by the seed.

4. A distance matrix between seeds is computed. The distance between two seeds is defined as the symmetrized Kullback-Leibler divergence between the two corresponding negative multinomial distributions (see Appendix B for the exact formula).

5. The distance matrix is used as input to the hierarchical clustering algorithm with average linkage, (the "hclust" function available in R is used) . Hierarchical clustering produces a tree where the leaves represent the different seeds, internal nodes represent clusters of seeds and where the height of an internal node represents the similarity between the two clusters being merged (see Figure 4.6).

6. The tree produced by the hierarchical clustering algorithm is cut at a distance from the root such that the resulting tree has exactly k nodes. Each node is a group of seeds that will be used to initialize the parameters corresponding to chromatin state.

4.2.6 Making ChIP-seq experiments comparable

Even if in most cases EpiCSeg can be used with raw count data, there are situations where this might not be appropriate and where a normalization procedure is required. This happens, for instance, when the same hidden Markov model is trained on multiple datasets, in order to have a comparable state annotation across datasets. In this setting the input data consists of a set of n_{rep} count matrices $\boldsymbol{c}^{(1)}, \boldsymbol{c}^{(2)}, ...\boldsymbol{c}^{(n_{\text{rep}})}$ where each matrix has n rows and m columns, representing the same set of m histone marks the same set of n bins. Each count matrix is derived from a separate dataset, which could be, for instance, a different cell type or the same cell type in two different conditions. A comparable state annotation across datasets can be useful, for instance, to test whether some regions are associated

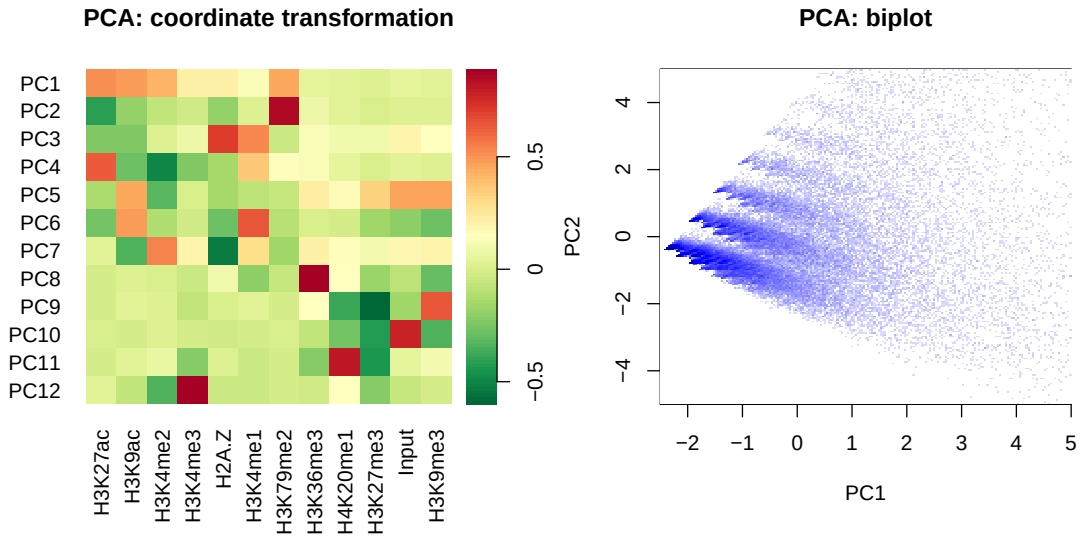

Figure 4.4: *PCA in the initialization algorithm. PCA transforms each observation vector into a vector where each element represents a principal component. The plot on the left shows the rotation matrix used to perform this transformation. The principal components are as many as the number of histone marks (in this example 12). The plot on the right shows a scatterplot of two principal components (all regularities in the scatterplot are due to the fact that the data is discrete). PCA is useful in determining the directions of variation and to discern the most meaningful trends in the data.*

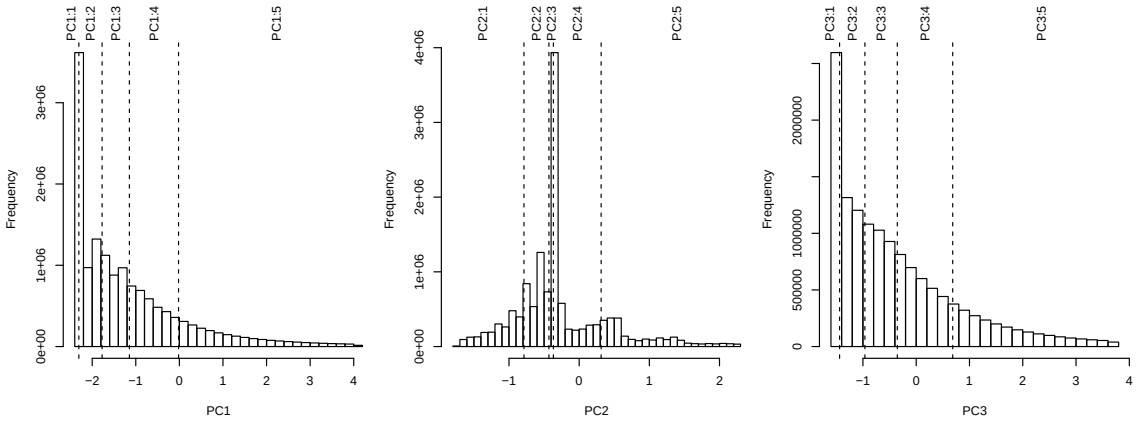

Figure 4.5: *Computation of seeds for the initialization algorithm. Each seed is a group of observations obtained by splitting a principal component into n_{lev} parts, so that each group of observations has approximately the same size. In this example $n_{lev} = 5$ and the separation is shown by the dashed lines. This results in n_{lev} seeds per principal component (5 times 12 in this example). For display purposes only the first principal components are shown, but the splitting is done for every principal component.*

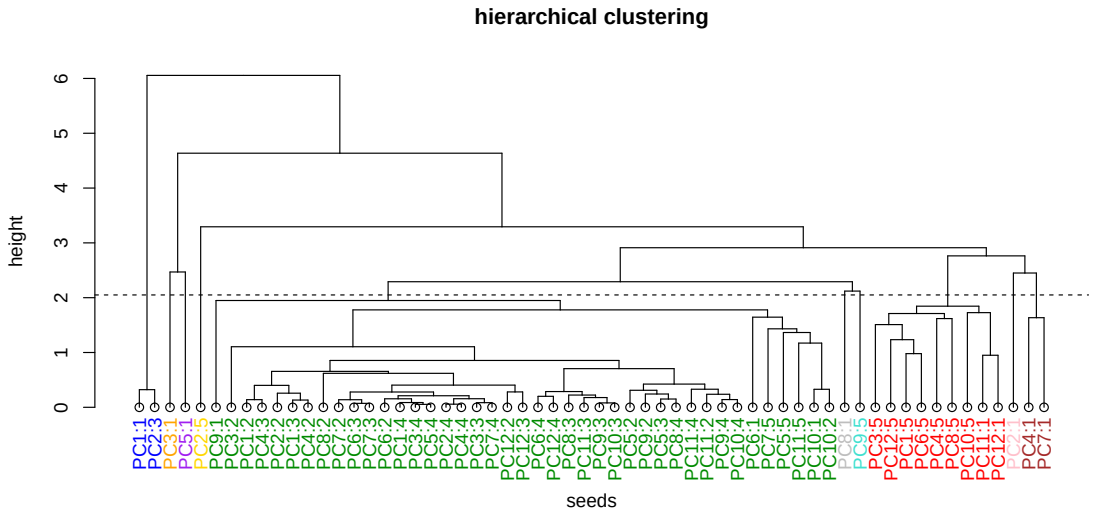

Figure 4.6: *Hierarchical clustering of seeds in the initialization algorithm. A pairwise distance between seeds is computed and is used for hierarchical clustering. The agglomeration of the leaves stops when k subtrees remain disjoint, where k is the desired number of clusters to initialize. In this case $k = 10$ and each subtree is represented by a distinct color of the leaves. The k subtrees can be used to initialize k different negative multinomial distributions.*

to different chromatin states. This would suggest that these regions are regulated differently.

In principle, EpiCSeg could treat these matrices as a single matrix with m marks and nn_{rep} bins and perform training as in the regular case. However in general there are systematic differences across samples that are not due to biological reasons and that would lead to wrong conclusions. The most important and best understood source of systematic differences is the total number of reads in each experiment. Let $\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, ...\boldsymbol{x}^{(n_{\text{rep}})}$ denote the vectors of length n corresponding to one of the m marks across the different dataset. If in the first dataset the ChIP-seq sample for this mark contains twice as many reads as in the second dataset, we would expect that for most of the bins $\boldsymbol{x}^{(1)} \sim 2\boldsymbol{x}^{(2)}$. There can be other sources of systematic differences that are less well-understood. For instance, referring to the ChIP-seq protocol outlined in Subsection 1.2.1, the size distribution of the DNA fragments in step 2 can be different across samples, or the separation of protein-bound DNA from the genomic background in step 3 can be more or less precise.

EpiCSeg approaches these problems by transforming the $\boldsymbol{x}^{(h)}$ vectors to corrected $\dot{\boldsymbol{x}}^{(h)}$ vectors, for $1 \leq h \leq n_{\text{rep}}$, and then uses the corrected input data as in the regular case. EpiCSeg offers two normalization procedures and it can be easily extended with user-defined functions. The first one, implemented in the "linearNormalization" function, consists in dividing each $\boldsymbol{x}^{(h)}$ vector by a number s_r called **size factor**. The corrected counts for bin i and sample h is simply $\dot{x}_i^{(h)} = \lfloor x_i^{(h)} s_r^{-1} \rfloor$. Size factors take into account how much each ChIP-seq sample has been sequenced

and they are estimated using the edgeR package [102]. The second normalization strategy, implemented in the "quantileNormalization" function, produces normalized $\dot{\boldsymbol{x}}^{(h)}$ vectors that share the same distribution. More precisely, let $\text{order}(\boldsymbol{x}^{(h)})_i$ denote the index where the i-th smallest element in $\boldsymbol{x}^{(h)}$ appears (draws are resolved randomly). After normalization $\dot{x}^{(h)}_{\text{order}(\dot{\boldsymbol{x}}^{(h)})_i} = y_i$ for all h and all i, where $\boldsymbol{y}$ is a reference count vector of length n with increasing values. The reference count vector is obtained from the $\boldsymbol{x}^{(h)}$ vectors:

$$y_i = \text{median}\{x^{(1)}_{\text{order}(\boldsymbol{x}^{(1)})_i}, x^{(2)}_{\text{order}(\boldsymbol{x}^{(2)})_i}, ...x^{(n_{\text{rep}})}_{\text{order}(\boldsymbol{x}^{(n_{\text{rep}})})_i}\}.$$

Note that these approaches are by no means a general solution, and therefore EpiCSeg does not compete with tools specifically designed to detect differences across ChIP-seq samples (see [102–106]). Nonetheless, running EpiCSeg on multiple datasets might still be useful for detecting large-scale differences between epigenomes and for providing a consistent state annotation.

4.2.7 Computational considerations

The update formulas 4.2 and 4.3 can be very costly to compute. The optimization function consists of a summation over all the n bins whose values depend on the f_{NB} function, which is a very costly function. Numerical methods need to evaluate the optimization function, or its derivative, a number n_r of times before returning the updated value for r_j, where typical values for n_r range from 10 to 30 iterations, and where these evaluations need to be done serially, i.e. they cannot be executed in parallel. Assuming that $n \sim 1.5 \cdot 10^7$ (as for a whole human genome and with a bin size of 200 bps) and $n_r \sim 20$, the Expectation Maximization algorithm would need to evaluate the f_{NB} function about $3 \cdot 10^8$ times at each iteration.

This problem can be alleviated by grouping together evaluations of the function $f_{\text{NB}}(c_{i+}; \mu_j^{(t+1)}, r_j)$ with the same c_{i+} value, similarly as for the mixture model algorithm presented in Subsection 3.2.6. Let

$$D = \{c_{i+} : 1 \leq i \leq n\},$$

and

$$\delta^{(t)}_{dj} = \sum_{i=1}^{n} \gamma^{(t)}_{ij} \cdot [c_{i+} = d],$$

where the expression delimited by square brackets evaluates to one when the expression inside it is true and to zero otherwise. Then the update formulas 4.2 and 4.3 can be rewritten respectively as

$$r_j^{(t+1)} = \arg\max_{r_j} \sum_{d \in D} \delta^{(t)}_{dj} \log f_{\text{NB}}(d; \mu_j^{(t+1)}, r_j),$$

and

$$r^{(t+1)} = \arg\max_{r} \sum_{d \in D} \sum_{i=1}^{n} \delta^{(t)}_{dj} \log f_{\text{NB}}(d; \mu_j^{(t+1)}, r),$$

respectively. Considering that D, even in genome-wide datasets, rarely contains more than 10^4 elements, the f_{NB} function now needs to be evaluated $1.5 \cdot 10^3$ times less frequently.

Assuming that the Baum-Welch algorithm needs n_i iterations to complete (which is typically around 100 iterations), we can estimate the time complexity of the whole training process (excluding initialization). At each iteration the algorithm needs to compute the probability of each observation of m counts given each of the k states, which takes time $O(nmk)$, then the forward-backward algorithm is used to compute the quantities $\gamma_{ij}^{(t)}$ and $\xi_{iuv}^{(t)}$, which takes time $O(nk^2)$, and finally these quantities are used to update the parameters of the model, which takes time $O(nmk)$ for the parameters of the multinomial distributions and $O(n_d n_r k)$ for the parameters of the negative binomial distributions, where n_d denotes the number of elements in D (both in the dependent and independent mode). This yields the following overall time complexity:

$$O(n_i(nk(n+k) + n_d n_r k)).$$

Also the time complexity of the initialization algorithm can be estimated. The initial PCA step requires time $O(nm^2 + m^3)$ (covariance matrix computation and eigenvalue decomposition). The computation of the seeds requires sorting the n values of each of the m principal component, which takes time $O(nm \log n)$. Finally, computing the distance matrix between all seeds and performing hierarchical clustering on them requires time $O(m(mn_{\mathrm{lev}})^2 + (mn_{\mathrm{lev}})^3)$. The overall asymptotic time complexity of the initialization algorithm is therefore:

$$O(nm(m + \log n) + (mn_{\mathrm{lev}})^3).$$

4.2.8 Implementation

EpiCSeg has been implemented as an R package with two main design goals in mind: ease of use and efficiency. The interface is simple and familiar to the large bioinformatics and statistics community using the R language. A command-line interface is also available, for those users not familiar with R. At the same time, the most time-consuming operations have been developed in C++, parallelized with OpenMP [107] and interfaced with R using the Rcpp package [108], which ensures efficiency and scalability with the number of cores in shared-memory architectures (see also Appendix C).

4.3 Results

As the chromatin segmentation problem is an unsupervised learning problem there is no clear performance score which can be used to compare segmentations by different methods. To make the comparison as fair and comprehensive as possible we adopted two strategies. First, we define and compute a number of performance indicators based on the association between chromatin states and validation data.

Second, we compare the different segmentations qualitatively, i.e. without using any performance indicator.

These comparisons also suggest alternative solutions to the task of interpreting the models provided by a segmentation algorithm. The sensitivity and precision scores used in the quantitative comparison show how validation data can be used to identify a state which most likely represents a given genomic feature. The genome-wide statistics used in the qualitative comparison show that each state has a peculiar distribution with respect to genes and a particular signature in terms of histone mark abundances which can be related to known biological processes.

The tools chosen for the comparisons are EpiCSeg and ChromHMM. Segway could not be included here because the time required for its training process is orders of magnitude larger and also because it works at single base pair resolution, while EpiCSeg and ChromHMM, as well as our validation procedure, use a binning scheme to reduce the high noise levels in the read counts.

To be able to draw relatively general conclusions, we compared the algorithms on 4 different datasets provided by the ENCODE consortium [70]:

- IMR90: lung fibroblast cells with 27 histone marks,
- H1: embryonic stem cells with 26 histone marks,
- K562_1: myelogenous leukemia with 11 histone marks and 1 control experiment,
- K562_2: same as above. The K562_1 and K562_2 datasets derive from an ENCODE dataset where two replicates per histone mark are available.

For each of these cell types ENCODE also provided a RNA-seq and DNase I hypersensitivity experiments that were used for validation, as described in the next section.

4.3.1 The supervised annotation

The aim of the supervised annotation is to provide a benchmark dataset to evaluate the efficiency of a segmentation algorithm. This subsection describes the procedure used to derive such an annotation.

The result of this procedure is the characterization of four different chromatin environments per cell line: **DNase+TSS** and **DNase-TSS**, characterized by a high DNase I HS signal and separated according to their proximity to an annotated TSS, **RNA**, characterized by a high RNA-seq signal, and **intergenic**, characterized by their long distance from RNA and DNase environments.

The input to the procedure is a RNA-seq and a DNase-I hypersensitivity experiment from the same cell type as the histone mark experiments used for the segmentation. Additionally the procedure uses the annotated transcripts available from the GENCODE database [109], version 19 (Ensembl 74). More precisely, all annotations of type "transcript" and of level 1 and 2 where considered. The binning scheme used for this procedure is the same as the one used for segmentation,

i.e. the bin have size 200 base pairs, with the difference that all bins overlapping a non-assembled region of the genome were discarded.

In the DNase-I hypersensitivity tracks, coming from single-end sequencing experiments, the accessibility signal per bin was defined as the number of reads whose 5' end maps within the bin. The top 2% bins were considered accessible and the top 0.8% were considered strongly accessible.

In the RNA-seq tracks, coming from paired-end sequencing experiments, regions ranging from the leftmost to the rightmost coordinate of a mapped read pair were treated as unstranded transcribed regions. The coverage per base pair was computed as the number of such transcribed regions that cover the base pair, and the transcription signal per bin was defined as the average coverage in the bin. The top 15% bins were considered transcribed.

The following criteria were used to define each chromatin environment:

- A **DNase** bin was defined as a strongly accessible bin. If the bin is closer than 500 base pairs to an annotated TSS it was considered a **DNase+TSS** bin, otherwise a **DNase-TSS** bin.

- A **RNA** bin was defined as a transcribed and non-accessible bin and such that all the 10 bins to the right and to the left are also transcribed and non-accessible.

- A bin was annotated as **intergenic** if that bin, as well as the 100 bins to the left and to the right, are neither a DNase nor a RNA bin.

4.3.2 Comparison with validation data

We ran ChromHMM and EpiCSeg genome-wide on the four datasets. The number of chromatin states was set to 10, the number of processing threads was set to 10, and all other parameters were set to their default values. In particular, both EpiCSeg and ChromHMM use the same binning scheme. The runtime of the two algorithms was similar: both tools performed genome-wide training and prediction in 15 to 35 minutes with neither method showing consistently shorter runtimes (see also Figure A.2).

We first measured how well an algorithm can recognize large regions with unusually low levels of histone marks. These regions are typical in genome-wide datasets due to mappability artifacts or low levels of chromatin accessibility, and it is desirable to safely exclude them. In this measurement, we identified the bins corresponding to assembly gaps, i.e. large regions of the reference genome where the sequence is not known and where no reads can be mapped. Next, we identified which state most likely represents assembly gaps by selecting the state with the highest precision. Given a state, the **precision** is the fraction of bins assigned to this state that are assembly gaps and the **sensitivity** is the fraction of assembly gaps that are assigned to this state. Figure 4.7 (a) shows that in all datasets and both in EpiCSeg and in ChromHMM almost all assembly gaps are annotated with the same state, however in EpiCSeg this state overlaps with the assembly gaps much more

precisely, especially in the K562_1 and K562_2 datasets. The fact that the precision always remains relatively low suggests that assembly gaps are not the only regions with unusually low levels of histone marks. Assembly gaps bins were excluded in the computation of all other performance indicators.

Next, we measured how well chromatin states can predict gene expression. For this purpose we used a cell-type specific RNA-seq experiment for each dataset. As a measure of gene expression levels we used the logarithm of the average RNA-seq coverage per bin (adding a pseudo-count of 1) and as a measure for predictive power we computed the $\boldsymbol{R^2}$ resulting from standard linear regression with a categorical predictor (the chromatin states). Figure 4.7 (b) shows that EpiCSeg and ChromHMM have a similar predictive power, but the former tends to perform better. The low R^2 values observed in the IMR90 and H1 datasets might suggest that in datasets with many ChIP-seq tracks the segmentation algorithms are less influenced by transcription-associated histone marks (e.g. H3K36me3).

For the next performance indicators we used the supervised annotation described in 4.3.1. Note that in this annotation some bins remain unannotated and they are not considered in the following.

Using the DNase+TSS bins as a gold-standard set of active TSSs we measured how well an algorithm can recognize active promoters. We selected the chromatin state that overlaps DNase+TSS bins with the highest precision. Figure 4.7 (c) shows that in all cases EpiCSeg identifies a chromatin state overlapping putative TSSs with a considerably higher precision than in ChromHMM. Often this state also overlaps more TSSs than in ChromHMM except in the H1 dataset (here, however, we could achieve a higher performance than with ChromHMM by merging two EpiCSeg states).

Next, we used the supervised annotation to compute an overall association score between external datasets (TSS annotation, RNA-seq and DNase I hypersensitivity) and chromatin states. As performance measure we used the **mutual information**, which can be estimated from a contingency table between the chromatin states vector and the chromatin environments vector. Figure 4.7 (d) summarizes the results and suggests that EpiCSeg is more strongly associated to the validation data.

To test the generality of our conclusions, we repeated the comparisons described above varying the number of states from 2 to 40 (see Figure 4.8). The results suggest that our conclusions are unlikely to depend on a particular choice of the parameters or on a particular initialization of the maximization algorithms. The results also show that the precision in assembly gap and TSS prediction, as well as the association to transcription and to the supervised annotation, tend to grow with the number of states, while the sensitivity in assembly gap and TSS prediction decreases, suggesting that the state representing a given genomic feature will eventually be split into two or more subtypes when increasing the number of states. A large number of states, however, renders the biological interpretation of the model difficult. The BIC and AIC methods [110], which determine the optimal number of states by penalizing the likelihood according to the number of parameters, failed in suggesting a number within the explored range (data not shown). We believe that such a choice should be a compromise between interpretability and accuracy of the

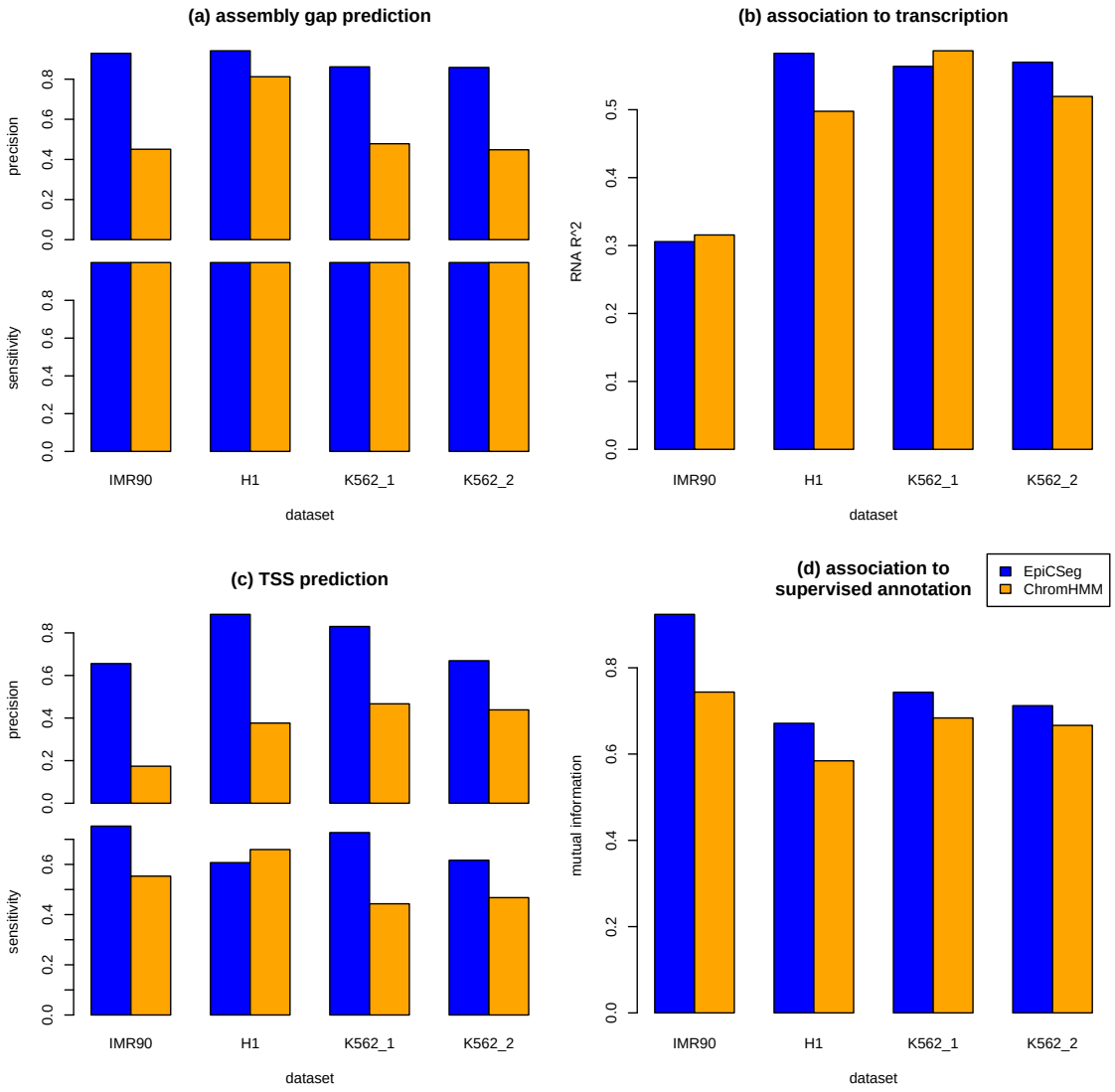

Figure 4.7: *Comparison with validation data. In (a) and (c) chromatin states are used as binary classifiers to predict assembly gaps and TSSs, respectively. The true positives are the bins annotated both with the feature and the given chromatin state. Precision and sensitivity are defined respectively as the number of true positives over the number of bins annotated with the state, and the number of true positives over the number of bins annotated with the feature. In (b) chromatin states are used to predict transcription levels. The value to be predicted is the log-transformed RNA-seq coverage per bin and the predictor is the chromatin state per bin. The R^2 values were computed using standard linear regression. In (d) the association between the unsupervised segmentation and the supervised annotation is measured. The manual annotation was produced using RNA-seq data, DNase I hypersensitivity data and a gene annotation, while the chromatin states were computed using only histone marks.*

model.

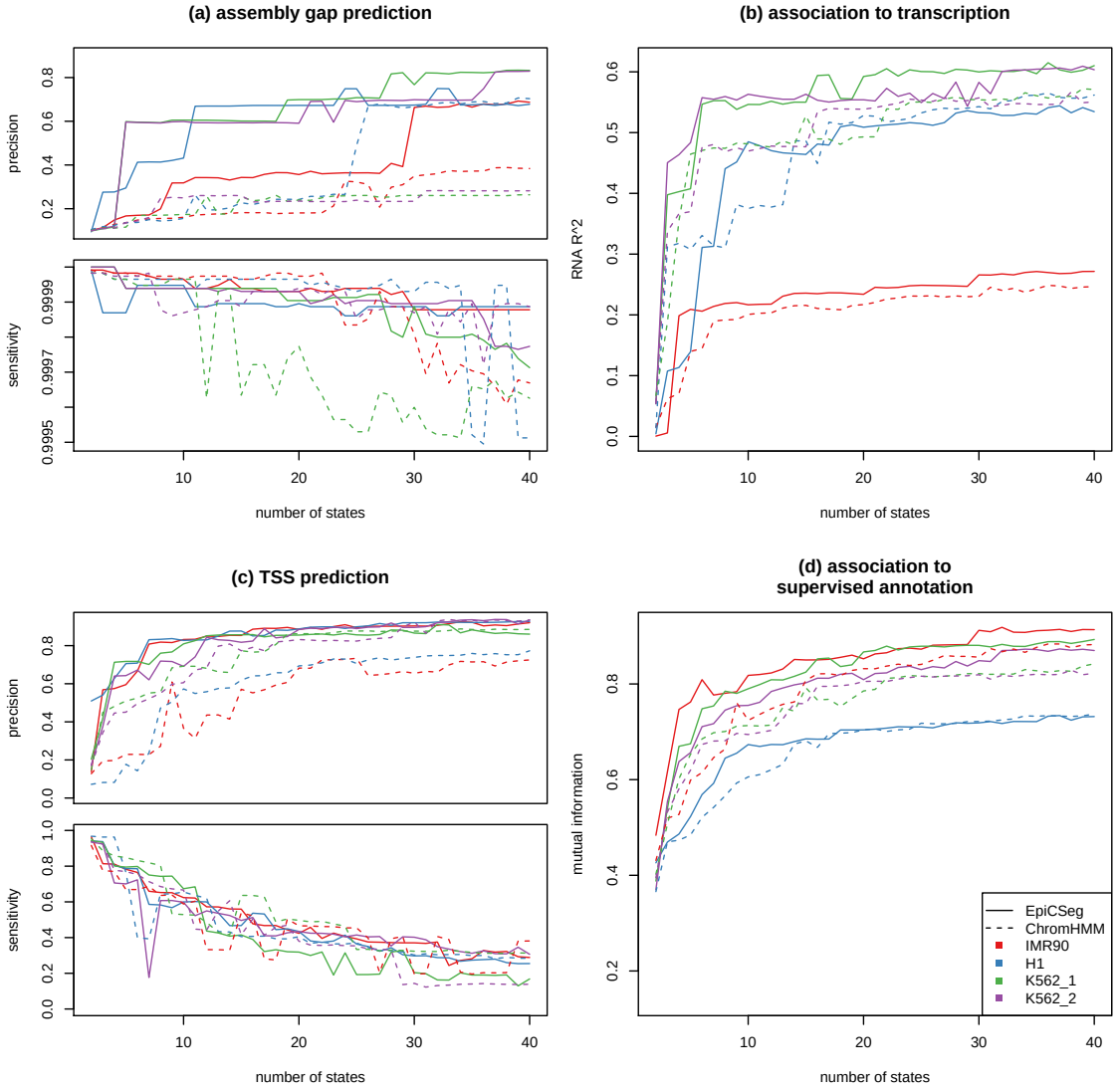

Figure 4.8: *Performance measures depending on the number of states. The same performance assessments as in Figure 4.7 are shown, but with a variable number of states (from 2 to 40) and using only the first human chromosome. Colors represent datasets, solid and dashed lines represent EpiCSeg and ChromHMM, respectively.*

4.3.3 A similarity measure between segmentations

To measure how similar two segmentations are to each other we used the **average Jaccard index**, which is a score between zero (completely different segmentations) and one (identical segmentations). The Jaccard index between two sets A and B is the ratio between the size of the intersection and the size of the union. If $|A|$ denotes the size of set A (in our setting, a number of base pairs), the Jaccard index can be

expressed by the formula $|A \cap B||A \cup B|^{-1}$. Given two segmentations, and assuming that there is a one-to-one correspondence between states, the average Jaccard index is computed as follows:

1. for each state s consider the two sets of base pairs I_1 and I_2, defined as the bins where state s occurs in segmentation 1 and 2 respectively,

2. measure the Jaccard index J_s between the two sets, defined as $J_s = |I_1 \cap I_2||I_1 \cup I_2|^{-1}$,

3. as the final score, consider the average Jaccard index J across all states $J = |S|^{-1} \sum_{s \in S} J_s$, where $|S|$ denotes the number of states,

4. choose the one-to-one correspondence between states as the one that maximizes the average Jaccard index.

4.3.4 Robustness comparison

We set out to test the algorithms' robustness to perturbations of the input data. The purpose of the first assessment is to test to which extent the chromatin states are influenced by technical variability, which includes sampling noise and differences in sequencing coverage. In fact this technical variability might affect EpiCSeg more than ChromHMM, as the former uses raw count data, while the latter uses normalized binary variables. The K562_1 and K562_2 datasets are suitable for this purpose because all samples come from the same cell type and replicate pairs are strongly correlated, even though there are considerable differences in sequencing coverage (see Supplementary Figures, Figure A.3). In order to have several measurements, we ran the segmentation algorithms (training and prediction) on each chromosome and each dataset separately and we computed the similarity between corresponding segmentations. As shown in the box plot in Figure 4.9 (left), the Jaccard indices obtained from EpiCSeg tend to be considerably higher than those obtained from ChromHMM, suggesting that the former tends to be more consistent across replicate datasets.

In the second assessment, we test the robustness of the algorithms to changes in the binning scheme. By default both algorithms (EpiCSeg and ChromHMM), bin the genome by assigning the first 200 base pairs of each chromosome to a bin, the second 200 base pairs to the next, and so on. Here, we studied to which extent the state assignment per base pair changes after shifting all bins by 100 base pairs. Figure 4.9 (right) shows that, for instance, in the K562_1 dataset with the EpiCSeg algorithm, on average more than 80% of the base pairs annotated with a certain state in one segmentation are annotated with the corresponding state also in the segmentation that uses the alternative binning scheme. Note, however, that part of this disagreement is simply due to the fact that the boundaries of two matching segments will necessarily differ by at least 100 base pairs. These results suggest that both algorithms are relatively robust to changes in the binning scheme, and that EpiCSeg tends to be more robust.

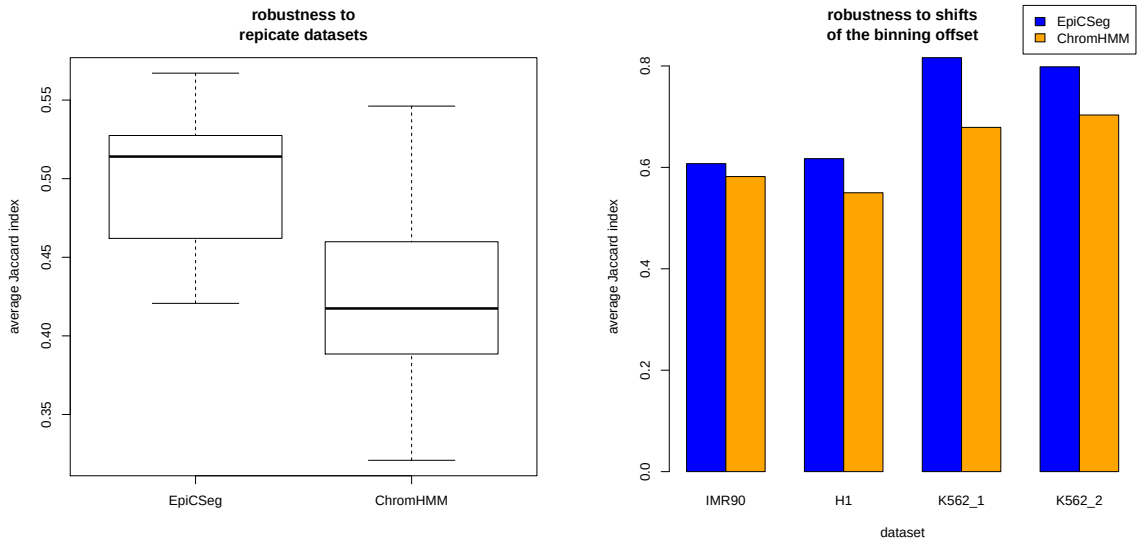

Figure 4.9: *Segmentation robustness. On the left, algorithms' robustness to replicate datasets. The K562_1 and the K562_2 datasets can be considered two perturbations of the same dataset. The segmentation algorithms were run independently on each chromosome on both datasets and the similarity between corresponding segmentations was measured. A higher Jaccard index means a greater robustness. The box plots show, among other things, the median (the thick line) and the first and third quartiles (the boundaries of the box) of the score distribution for each algorithm. On the right, algorithms' robustness to shifts of the binning offset. Reads have been counted using two different binning schemes. The segmentation algorithms were run using both schemes and the similarity between segmentations were measured.*

4.3.5 Qualitative comparison

In order to show the salient differences between the two algorithms without focusing on single regions, we collapsed the segmentation data into genome-wide summary statistics. The first summary statistic is a bar plot where each bar corresponds to a chromatin state and where its length is proportional to the state frequency. Additionally, edges between states of the two segmentations have been drawn with widths proportional to the number of overlapping bins. Another statistic shows where each state tends to localize with respect to genes. More precisely, for each annotated transcript in the GENCODE database [109] and for a given segmentation we considered a region comprising the transcript, 5000 bps upstream the TSS and 5000 bps downstream the TES, we labeled each base pair with its inferred state, and we rescaled the region between TSS and TES to a reference length. Finally, taking into account all transcripts, we counted how many regions are annotated with a given state at a given position. The third summary statistic is a heatmap showing the log-transformed average histone modification levels per state.

From Figure 4.10 (bottom left), we notice that both segmentations on the K562_1 dataset are strongly dependent on the genomic context, that is, they can capture and represent the most important biological processes acting on the chromatin. The clearest signals are a state peaking exactly at the TSS of the genes and a state which appears mainly in the body of the transcripts and peaks at the TES.

However, there are also some differences. The most apparent is that in the ChromHMM segmentation there is a state accounting for more than half of all bins, while the state distribution in EpiCSeg's segmentation is more balanced. This background state in ChromHMM is likely to be an artifact of the discretization step and to correspond to bins where most of the read counts are below the discretization threshold rather than to represent a well-defined chromatin state. The same background state corresponds mainly to three EpiCSeg states. One of them is associated to very low read counts for all marks. The analysis in Figure 4.7 (a) showed that almost all assembly gaps are annotated with this state and that they make up almost half of it. The other two states correspond to repressive chromatin environments enriched respectively with H3K27me3 and H3K9me3. The second apparent difference is that that the promoter state in EpiCSeg's segmentation is more tightly centered on the TSS, which is also reflected in the higher classification score observed in the performance comparison (Figure 4.7 (c)). These conclusions are confirmed also in the other three datasets (see Figures A.4, A.5 and A.7).

However, there are also some differences. The most apparent is that in the ChromHMM segmentation there is a state accounting for more than half of all bins, while the state distribution in EpiCSeg's segmentation is more balanced. This background state in ChromHMM is likely to be an artifact of the discretization step and to correspond to bins where most of the read counts are below the discretization threshold rather than to represent a well-defined chromatin state. The same background state corresponds mainly to three EpiCSeg states. One of them is associated to very low read counts for all marks. The analysis in Figure 4.7 (a) showed that almost all assembly gaps are annotated with this state and that they make up almost

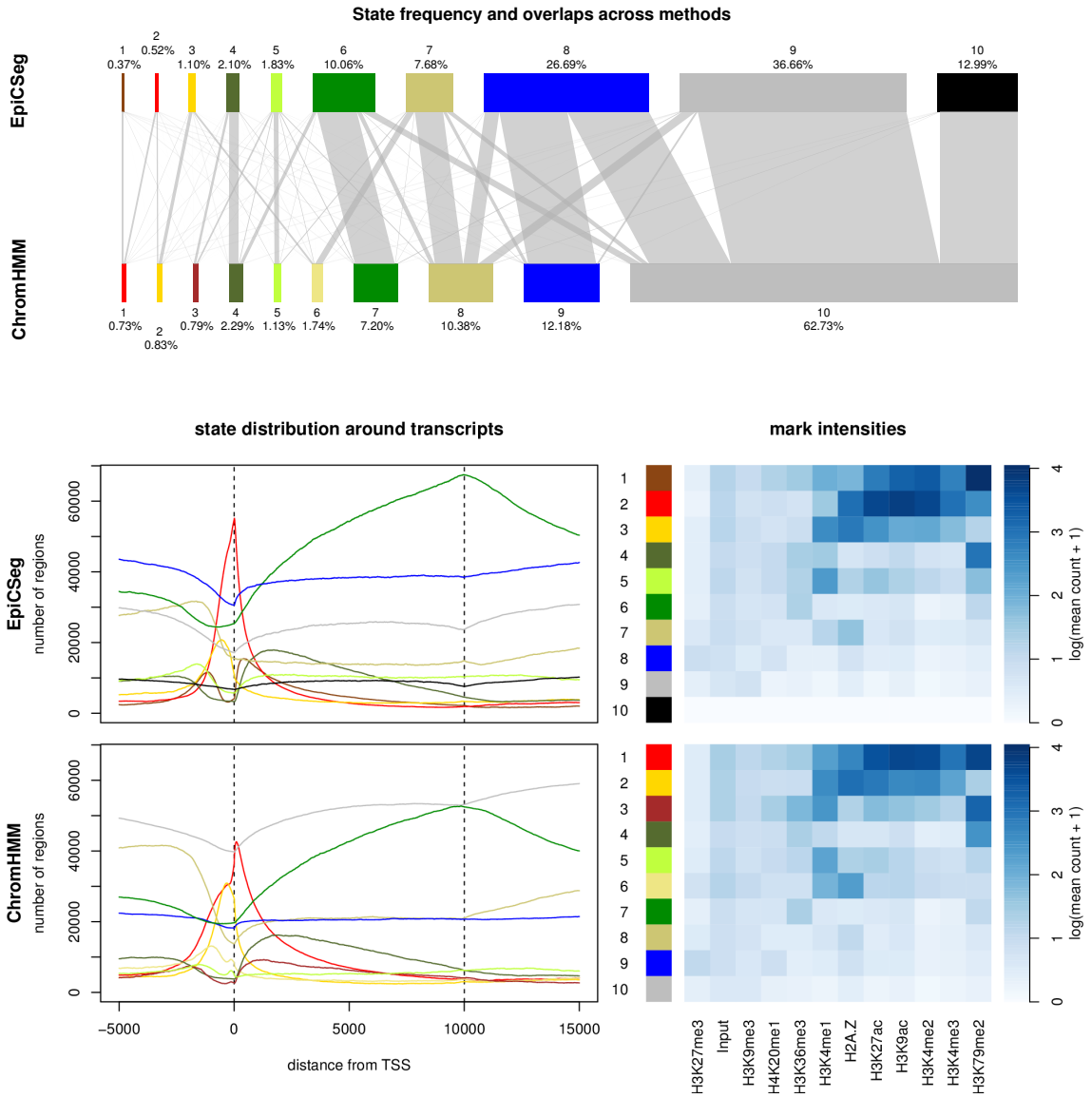

Figure 4.10: *Qualitative overview of EpiCSeg's and ChromHMM's segmentations. The plot at the top shows the frequency of each state in each segmentation as bars with variable widths. Additionally the edges' widths show the overlap between states of different segmentations. The two plots at the bottom left show how often a particular state occurs at a particular position of the transcript, rescaling all transcripts so that they have the same length. The two plots at the bottom right show the mark intensities depending on the state. The choice of the colors is arbitrary. All statistics were computed in the K562_1 dataset.*

half of it. The other two states correspond to repressive chromatin environments enriched respectively with H3K27me3 and H3K9me3. The second apparent difference is that that the promoter state in EpiCSeg's segmentation is more tightly centered on the TSS, which is also reflected in the higher classification score observed in the performance comparison (Figure 4.7 (c)). These conclusions are confirmed also in the other three datasets (see Figures A.4, A.5 and A.7).

Other smaller differences can be observed in Figure 4.10. For instance, EpiCSeg separates promoter-proximal regions into those with the known set of promoter-associated marks, H3K27ac, H3K9ac, H3K4me2-3 (state 2), and those with lower levels of promoter-associated marks and a very high level of H3K79me2 (state 1), whereas ChromHMM does not make this distinction (state 1). Furthermore, the state in ChromHMM with the highest levels of H3K4me1 (probably representing enhancer regions) is very similar to the promoter state considering its localization and the marks intensities, while in EpiCSeg there is a greater separation (between state 3 and state 2). These two last differences, however, cannot be always generalized to the other datasets.

4.3.6 Uncertainty in state assignments

We explored how confidently EpiCSeg's probabilistic model can assign a bin to a chromatin state in relation to the read coverage in the bin. As a measure of uncertainty in the state assignment we computed the posterior entropy per bin, which is the entropy of the probability distribution describing how probable each state is for that bin. The read coverage per bin is the sum of the read counts across all histone marks. The results of this explorative analysis can be seen in Figure 4.11 for the K562_1 dataset, which shows (i) a smoothed scatterplot of the posterior entropies versus the read coverage and (ii) the mean posterior entropy per read coverage level. The most apparent trend is that most of the entropies tend to cluster around 0, or to a much smaller extent, around 1, suggesting that for most of the bins the probabilistic model is very certain of the state assignment, or it is undecided between 2 alternatives. The second apparent trend is that the bins that can be most confidently classified are either bins with no reads at all, typically corresponding to assembly gaps, or bins with a very large number of reads, typically located in promoter regions, while bins with a read coverage between 10 and 100 are harder to classify. The same analysis performed in the other datasets leads to similar conclusions (data not shown). To summarize, EpiCSeg's model tends to be very certain of state assignments, with a weak dependence on the read coverage. This suggests that modeling the read counts directly does not necessarily introduce a high level of uncertainty in the state inference for bins with low-counts and supports our claim that EpiCSeg allows for assigning chromatin states to a larger portion of the epigenome compared to existing approaches.

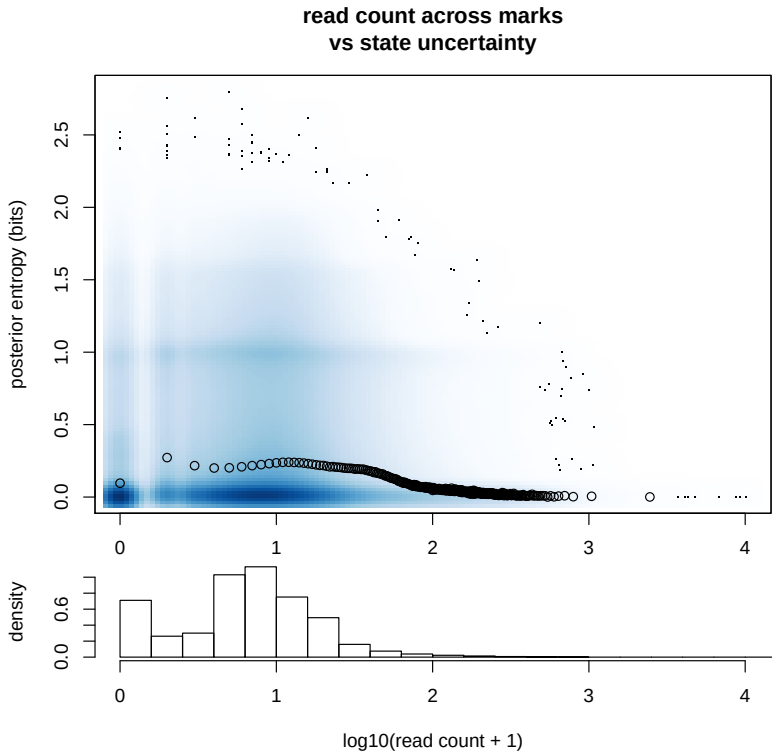

Figure 4.11: *Uncertainty in the state assignments. The smoothed scatterplot consists in the blue shades, which show the density of points for each combination of read coverage and uncertainty, and the small black dots, which are the outliers. Additionally, entropies have been averaged over groups of bins with the same or a similar read count, so that each group consists of at least 500 bins. The empty black circles show the average entropy level per group.*

4.4 Discussion

In genomic regions with similar functions, similar combinations of histone marks can be observed. These patterns in the histone marks abundances are called chromatin states. EpiCSeg is an algorithm that can learn the most important chromatin states by analyzing several ChIP-seq experiments simultaneously and assigns each genomic region to a chromatin state. Similarly to the ChromHMM algorithm, EpiCSeg divides the genome into consecutive bins and assumes a hidden Markov model to learn and infer the hidden sequence of chromatin states. In contrast to its predecessor, EpiCSeg's input data are natural numbers instead of binary variables, which has two important practical advantages. First, no arbitrary thresholds on the read counts are needed to decide when a mark is present or not, since the read counts can be directly used as input data. Second, because the input data contains more information than the binary variables, EpiCSeg segmentation is more accurate and can annotate a larger portion of the genome. Moreover, these advantages do not come at the expense of an increased runtime. In summary, EpiCSeg marks a considerable improvement upon its predecessors and paves the way for a more quantitative analysis of chromatin states.

The chromatin segmentation problem, however, is far from being solved. One important open issue is a more objective definition of chromatin state. If unsupervised learning methods are very useful for data exploration, especially when little is known about the patterns present in the data, they are suboptimal when used for the characterization of patterns that are known and expected. Today researchers expect from histone modification data a reliable annotation of certain genomic elements, such as promoters of transcribed genes, transcribed gene bodies and enhancers, and future research might focus on supervised or semi-supervised learning approaches for this purpose.

Another important research direction might address the problem of making chromatin states comparable across datasets. The solutions described in Subsection 4.2.6 attempt to correct the raw count data before segmentation and make strong assumptions. More principled approaches might account for differences between samples with a more accurate model during segmentation, rather than in a preprocessing step. Finally, it is not clear whether chromatin segmentation can be used for a more accurate identification of differences between datasets.

Chapter 5

Footprint discovery

This chapter presents Footifind: an algorithm that uses ChIP-exo data to learn the precise peak shape and the DNA sequences that characterize transcription factor binding sites. This method can be used to discover motifs, study the structure of the DNA-protein binding, and to detect putative binding sites.

5.1 Motivation

Transcription factors (TFs) play a fundamental role in gene regulation and cell differentiation. By binding to specific genomic locations, called binding sites, they can interact directly or indirectly with the transcription of one or more genes and thereby regulate their activity. Transcription factors are so important that the identity of a cell in an organism with different cell types can be explained to a large extent in terms of presence or absence of transcription factors [111]. In order to understand how TFs regulate the transcriptional program of a cell, it is important to know to which genomic locations each TF binds.

TF binding sites are often characterized by a short pattern in the DNA sequence called motif, which is fundamental for the TF to correctly recognize its target binding sites. Therefore, once the motif for a particular TF is known, a candidate set of binding sites can be obtained. However, the motif alone is not sufficient to precisely locate all binding locations. Other processes, such as the accessibility of the DNA, or other proteins competing or cooperating with the TF, can largely influence TF binding.

The ChIP-seq protocol [40] measures protein binding for a given protein, in a given cell sample, and at all possible genomic locations (see Section 1.2). The data provides a read count for each base pair in the genome, and binding sites can be recognized from clusters of high read counts, called peaks. However, these peaks are too broad to mark the binding site precisely. A putative ChIP-seq peak typically spans 200 base pairs, making it hard to understand where exactly the TF is bound and which motif it recognizes.

The ChIP-exo protocol [41] is a variant of ChIP-seq that achieves a greater resolution. With ChIP-exo, by using motif information to precisely define binding

sites, a common, precise pattern in the read counts emerges which is much more informative than a ChIP-seq peak and it is related to the way the TF is physically bound to the DNA. These qualitative observations lead to the concept of a footprint: a precise peak shape occurring at TF binding sites, similar to the footprints observed in DNase-I hypersensitivity data [112, 113].

Footprints are important for several reasons. First, they allow a more precise identification of TF binding sites. Second, these refined regions can be used to better understand the sequence preferences of the TF and to possibly discover new motifs. Third, footprints can provide structural information about the protein-DNA interaction. The detailed analysis of a footprint can suggest, for instance, whether a TF is binding as a monomer or as a dimer and how large its DNA binding domain is.

Previous computational approaches have focused on discovering sequence motifs from sequences obtained from ChIP-seq peaks [114–118], or on using known motifs to model TF footprints in DNAse I hypersensitivity data [55, 119, 120]. There are also methods that attempt to find binding sites, footprints and motifs simultaneously from ChIP-exo data [121], but this is achieved by a sequential combination of a peak-calling and a motif-discovery method rather than by an integrated understanding of sequence-level and count-level features.

Here we present Footifind: an algorithm that models TF binding as the simultaneous occurrence of a motif and a footprint and that infers such an integrated model without prior knowledge. Experimental results show that our approach is able to detect motifs supported by the literature and footprints compatible with the molecular structure of the TF. Moreover, Footifind can distinguish different categories of binding sites, which can be due to the same TF binding in different ways or to the binding of other unexpected proteins.

5.2 Methods

5.2.1 The ChIP-exo protocol

ChIP-exo provides reads that are more tightly centered around a TF binding site than with ChIP-seq. This experimental procedure is similar to the ChIP-seq protocol (see Subsection 1.2), with the addition of one step, called exonuclease digestion, that takes place after immunoprecipitation of the DNA fragments (step 3) and before purification (step 4). At this stage, the double-stranded protein-bound DNA fragments are incubated together with an enzyme, called lambda exonuclease, that digests each strand of a fragment proceeding in the 5'-to-3' direction (see Figure 5.1). What prevents the DNA from being completely digested are the crosslinks between protein and DNA, formed in step 1 of the ChIP-seq protocol. In fact each crosslink protects a region of about 5 base pairs upstream and downstream the crosslinked nucleotide. After digestion, the 5' end of a bound fragment coincides with the boundary of the protected DNA region while the 3' end remains intact, so that the fragment is long enough for sequencing and mapping.

After that ChIP-exo reads are mapped to the reference genome, it is convenient to consider the count signal generated by the 5' mapping positions of the reads. At TF binding sites, a peak pair is typically observable, where a peak in the sense strand is followed by one in the antisense strand (if the sense strand is oriented from left to right). This shape, however, is highly dependent on the crosslinks between DNA and TF. The distance between the two peaks, in fact, depends on the length of the protected DNA, on the number of crosslinks and on their strength. Moreover, because crosslinks can be more or less stable, the boundaries of the protected region can be more or less sharp, or there might be primary and secondary boundaries.

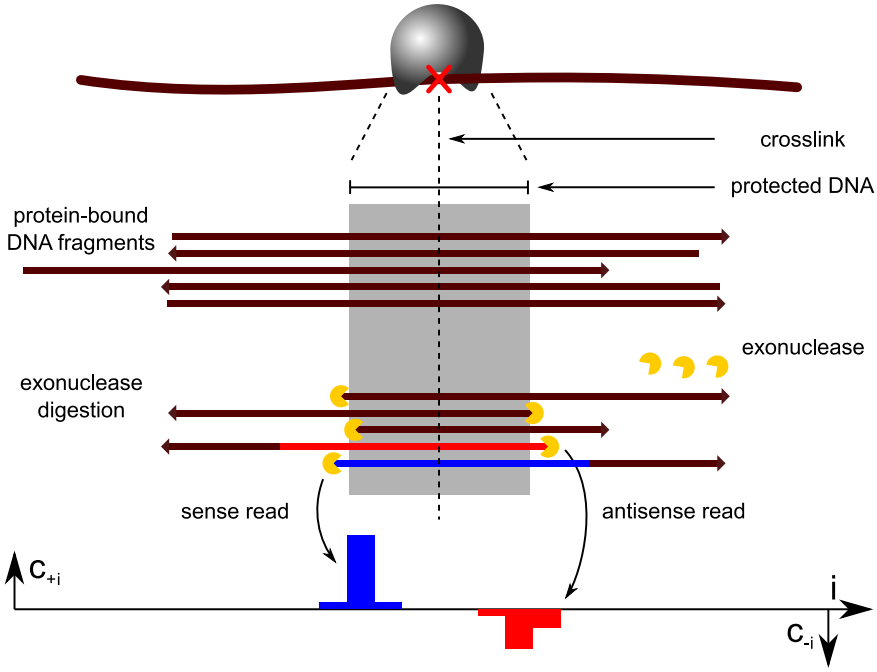

Figure 5.1: *Exonuclease digestion and the ChIP-exo count signal. Above, a cartoon of a TF bound to double-stranded DNA. Below, single-stranded DNA fragments relative to the TF above (the arrow denotes the 3' end). The shaded rectangle shows the DNA region that is protected by the crosslink. Exonuclease digestion progresses from the 5' to the 3' end of a DNA fragment and stops when it encounters the protected DNA region. In the count signal obtained by counting the 5' end of the reads, a peak pair is visible at the boundaries of the protected DNA region.*

5.2.2 Notation

We will use the index i to denote a position in the genome. The index i is a simple integer that encodes a chromosome, a number of base pairs from the beginning of the chromosome, and a genomic strand. The chromosome is encoded by concatenating all chromosomes together and counting the number of base pairs from the start of this artificial genome. The strand is encoded by using negative indices, so that positive

and negative values denote the forward and the reverse strand in the reference genome assembly, respectively (position 0 is undefined). The sense strand in relation to a position i refers to the strand encoded by index i, and the antisense strand to the other one.

The symbol s_i, where $s_i \in \{\mathtt{a}, \mathtt{c}, \mathtt{g}, \mathtt{t}\}$, denotes the reference genome sequence at position i. This implies that s_{-i} is the complementary base of s_i. The symbol c_i denotes the number of reads of a ChIP-exo experiment such that the 5' end of the read maps at position i. Reads mapping to the forward and reverse strand, therefore, increment c_i for positive and negative values of i, respectively.

5.2.3 A closer look at footprints

To clarify the concept of a footprint, we show a concrete example based on a ChIP-exo experiment for the TF FOXA1 in the human cell line MCF7 [122]. In this example, $n_p = 4275$ putative binding sites were obtained by using the known sequence motif of FOXA1 and by intersecting motif occurrences with regions with high read counts. Each putative binding site k has a start coordinate i_k and an end coordinate $i_k + w - 1$, where w is the window size. Figure 5.2 shows two representations of the corresponding count data. In the heatmap representation, each row corresponds to a binding site k with $1 \leq k \leq n_p$ and each column to a position j with $1 \leq j \leq w$. The blue and red squares at row k and column j represent the sense read count c_{i_k+j-1} and the antisense read count $c_{-(i_k+j-1)}$, respectively. The heatmap clearly shows that read counts at putative binding sites tend to follow a common pattern. A footprint can be more conveniently represented by computing the average peak shape across binding sites. The blue and red lines show for each position j the average read count $(n_p)^{-1} \sum_{k=1}^{n_p} c_{i_k+j-1}$ and $(n_p)^{-1} \sum_{k=1}^{n_p} c_{-(i_k+j-1)}$, respectively.

Figure 5.2 also shows the properties that characterize a footprint. First of all, footprints are highly position-dependent, i.e. the counts in a certain position of the window can be very different from those in the next. This typically results in sharp peaks in the average peak shape. Second, for each peak in one strand of the average peak shape, another corresponding peak should be observed in the other strand. More precisely, orienting the average peak shape from the 5' to the 3' end of the sense strand, each peak in the sense strand should be followed by a peak in the antisense strand. Note, however, that these peak pairs do not need to be symmetric. In Figure 5.2, for instance, the peak in the antisense strand is much sharper than the one in the sense strand, which looks more like two smaller peaks. Another important expectation is that each footprint should be related to a motif. Even though, in theory, it cannot be excluded that mechanisms other than a motif can lead to such information-rich peaks, this seems to be often the case, and our algorithm will leverage this property to discover footprints.

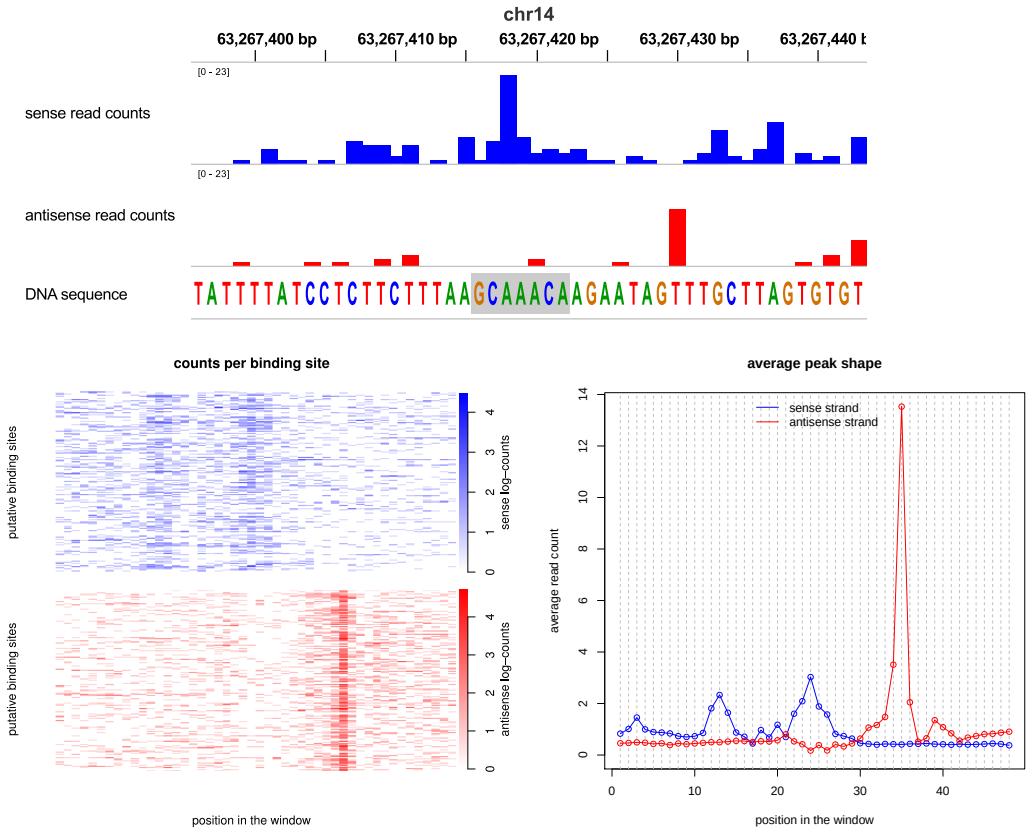

Figure 5.2: *Example of a footprint for TF FOXA1. On top, a putative binding site. Putative binding sites were obtained by considering genomic windows with high read counts and with an occurrence of the FOXA1 motif (shown in gray). On the bottom left, heatmap representation of a footprint. A large number of binding sites have been precisely aligned relative to the FOXA1 motif occurrence. Each cell of the heatmap shows the read counts for a particular binding site at a particular position of the window. On the bottom right, representation of a footprint as an average peak shape. The lines were derived by averaging the read counts shown in the heatmaps across all binding sites.*

5.2.4 Goals

The goal of the method is to characterize binding events in a ChIP-exo experiment in three ways:

1. by a motif,
2. by a footprint,
3. by the coordinates of the putative binding events.

The combination of a motif and a footprint will be referred to as a **footif**. In cases where TF binding sites exhibit considerably different footifs, we would like to obtain a motif, a footprint and a set of coordinates for each footif.

5.2.5 An iterative algorithm

Footifind's input data are the counts c_i for a ChIP-exo experiment and the bases in the reference sequence s_i, as defined in Subsection 5.2.2. The most important parameter is the window size, denoted as w, which specifies the number of base pairs that need to be considered to have a full description of a footif (by default $w = 100$). In general not every genomic position will be considered. Typically, regions with low ChIP-exo reads are filtered out beforehand, so as to reduce the runtime and to improve the performance. The set of valid start positions for a window representing a binding site, which is a user-defined parameter, will be denoted as I. In other words, the counts c_k and c_{-k} and the sequence s_k will be considered only if there is a start position i in I such that $i \leq k < i + w$.

The overall strategy of Footifind, shown in Figure 5.3, consists in iterating between two main steps: fitting the footprint and the motif models from a set of candidate binding sites (the training step), and deriving a refined set of candidate regions from the learned models (the scoring, score integration and thresholding steps). At the first iteration the candidate regions are those with the highest read counts. The iterations stop when there is no more considerable change in the learned models. This results in a set of candidate binding sites associated to a footif. To detect more than one footifs, the whole procedure is started again after removing the candidate binding sites related to the previous footifs from the input data.

5.2.6 Model-based classifiers

Footifind is based on a footprint and a motif classifier. Each classifier is used to compute a score for each possible window, and the score is used to discern between binding events and background regions. Moreover, each classifier is model-based, which means that the score is derived from two probabilistic models: a peak and a background model. The former assumes that the data in the window originates from a binding event, while the latter assumes that the window represents background noise, or signal due to footifs other than the one being scanned, or not exactly the start of the binding site. Lastly, the score is derived from the log-ratio between the

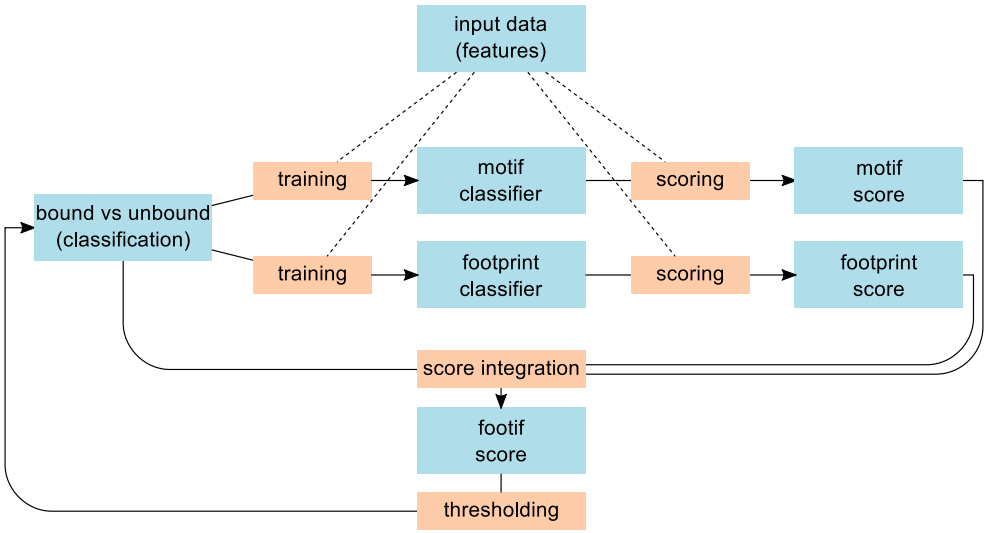

Figure 5.3: *Footifind's workflow. Each iteration starts with a classification of each possible window as a bound or unbound site. Next, two linear classifiers are trained: one based on the sequence (the motif classifier), the other based on the counts (the footprint classifier). The classifiers are used to score all possible windows. Next, these scores are integrated by normalizing the score distributions and by learning the relative importance of the counts with respect to the sequence. Finally a new set of putative binding sites is obtained by selecting the top scoring non overlapping windows. Upon convergence, the putative binding sites are removed and the whole procedure is started again (not shown).*

two probabilities and it can therefore be regarded as a näive Bayes classifier. The next two subsections will present the two classifiers in more detail.

5.2.7 The footprint classifier

The footprint classifier is a scoring procedure that associates a score $z_{\mathtt{f}i}$ to each index $i \in I$ based on the sense read counts $c_i, ...c_{i+w-1}$ and the antisense read counts $c_{-i}, ...c_{-(i+w-1)}$. This subsection not only describes how the score is computed, but it also contains some considerations about the implementation choices.

The scoring method is based on two generative probabilistic models for the read counts: a peak and a background model. In formulas, the model assumes that there are two random vectors $\boldsymbol{C}^{(\mathtt{p})}$ and $\boldsymbol{C}^{(\mathtt{b})}$, with components

$$\boldsymbol{C}^{(j)} = \left(C_1^{(j)}, C_2^{(j)}, ...C_w^{(j)}, C_{-1}^{(j)}, C_{-2}^{(j)}, ...C_{-w}^{(j)}\right), \text{ for } j = \mathtt{p} \text{ and } j = \mathtt{b},$$

and that the counts relative to a window starting at position i are a realization of one of the two random vectors:

$$\boldsymbol{C}^{(j)} = (c_i, ...c_{i+w-1}, c_{-i}, ...c_{-(i+w-1)}), \text{ for } j = \mathtt{p} \text{ or } j = \mathtt{b}.$$

The choice of the appropriate probability distribution is, therefore, a key factor for the accuracy of the classifier. One desirable characteristic of the probability distribution is that the components of the $\boldsymbol{C}^{(j)}$ vector should be correlated. In case the window represents a binding event, it is reasonable to assume that the read counts vary proportionally to the extent of the protein-DNA binding, leading to a positive correlation coefficient between positions of the window. Also in case the window is in a background region it is reasonable to assume that noise levels at adjacent positions should be correlated. Another desirable feature is that the variance of each component C_i should be considerably higher than the mean, as this is a property frequently observed in count data. Figure 5.4 suggests that these intuitions apply to the putative binding sites described in Figure 5.2.

One possibility is to model the random variables $\boldsymbol{C}^{(\mathtt{p})}$ and $\boldsymbol{C}^{(\mathtt{b})}$ as multivariate random gaussians. This approach has already been applied to count data for a similar classification problem [123]. However, in our context, the gaussian assumption seems inadequate, because the features c_i are discrete and typically concentrated around 0. Moreover, the number of parameters required to characterize a multivariate gaussian distribution is quadratic in the window length w, which might make the training step challenging. In Footifind $\boldsymbol{C}^{(\mathtt{p})}$ and $\boldsymbol{C}^{(\mathtt{b})}$ are assumed to follow a negative multinomial distribution (see Subsection 2.2.3), which is consistent with the qualitative observations made above. This approach has also been taken in Pique-Regi *et al.* [55] for the analysis of footprints in DNase hypersensitivity data.

In formulas, the footprint classifier assumes that:

$$\boldsymbol{C}^{(j)} \sim \mathrm{NM}(\mu^{(j)}, r^{(j)}, \boldsymbol{p}^{(j)}), \text{ for } j = \mathtt{p} \text{ and } j = \mathtt{b}.$$

Note that the $\mu^{(j)}$ parameter can be interpreted as the overall strength of the binding event (or of the noise level) in the window, and the $\boldsymbol{p}^{(j)}$ parameters as coefficients

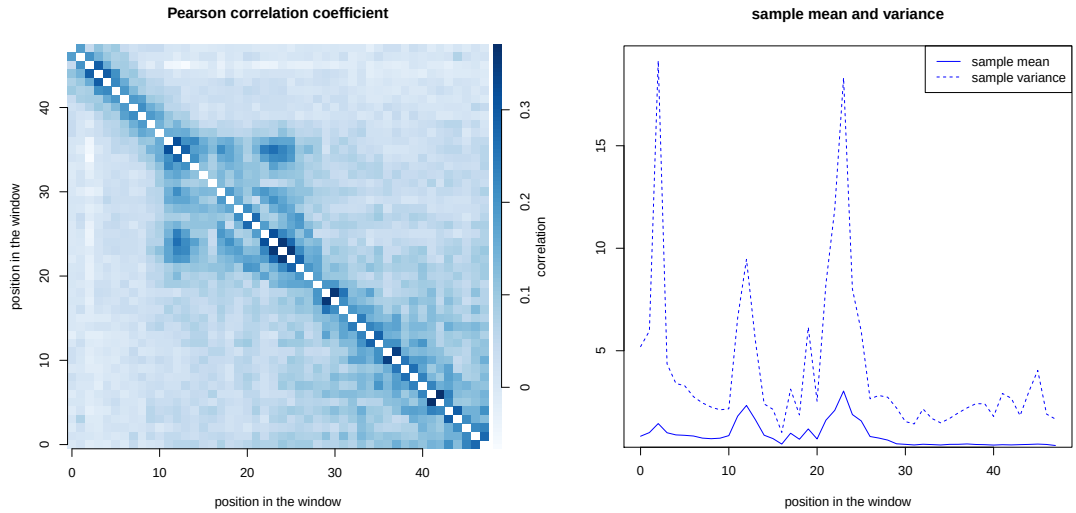

Figure 5.4: *Correlation and overdispersion in the components of a footprint. The read counts for FOXA1 putative binding sites were derived as described for Figure 5.2, which results in a count matrix with rows as binding sites and columns as positions of the footprint. On the left, the Pearson correlation coefficient between every two columns of the matrix is shown. Most correlations are positive, supporting the intuition that the counts vary proportionally to the extent of the protein-DNA binding (the coefficients on the diagonal, which equal 1 by definition, have been set to 0 for display purposes). Right, the sample mean and sample variance for each position of the matrix is shown. The variance is considerably higher than the mean, supporting the claim that a Poisson model cannot account for this degree of variation.*

of proportionality between the overall read count and the read count at position i of the window.

The parameters are learnt from a classification of the windows in I into peaks and background: $I = I^{(\mathtt{p})} \cup I^{(\mathtt{b})}$. For each set of windows, maximum likelihood estimation is performed using the procedure outlined in Subsection 4.2.4.

The footprint score for a certain window i, denoted $z_{\mathtt{f}i}$, is the log-ratio between the probability of the two conditions:

$$z_{\mathtt{f}i} = \log \frac{\mathrm{P}\left\{\boldsymbol{C}^{(\mathtt{p})} = (c_i, c_{i+1}, ...c_{i+w-1}, c_{-i}, c_{-(i+1)}, ...c_{-(i+w-1)})\right\}}{\mathrm{P}\left\{\boldsymbol{C}^{(\mathtt{b})} = (c_i, c_{i+1}, ...c_{i+w-1}, c_{-i}, c_{-(i+1)}, ...c_{-(i+w-1)})\right\}}.$$

5.2.8 The motif classifier

Similarly as for the footprint classifier, the motif classifier computes a score $z_{\mathtt{m}i}$ for each index $i \in I$ based on the sequence $s_i, s_{(i+1)}, ...s_{(i+w-1)}$. This is achieved by assuming that the sequences occurring at peaks and those occurring in background regions are independent realizations of two random sequences $\boldsymbol{S}^{(\mathtt{p})}$ and $\boldsymbol{S}^{(\mathtt{b})}$, where $\boldsymbol{S}^{(j)} = (S_1^{(j)}, S_2^{(j)}, ...S_w^{(j)})$ for $j \in \{\mathtt{p}, \mathtt{b}\}$, and that each random sequence is characterized by a position probability matrix (see Subsection 2.2.1):

$$\mathrm{P}\left\{\boldsymbol{S}^{(j)} = (s_i, s_{(i+1)}, ...s_{(i+w-1)})\right\} = \prod_{k=1}^{w} \sum_{\alpha \in \{\mathtt{a},\mathtt{c},\mathtt{g},\mathtt{t}\}} q_{k\alpha}^{(j)} \cdot [s_{i+k-1} = \alpha].$$

The parameters $\boldsymbol{q}_k^{(j)}$ are computed by maximum likelihood estimation as for the footprint classifier and the score is defined as:

$$z_{\mathtt{m}i} = \log \frac{\mathrm{P}\left\{\boldsymbol{S}^{(\mathtt{p})} = (s_i, s_{(i+1)}, ...s_{(i+w-1)})\right\}}{\mathrm{P}\left\{\boldsymbol{S}^{(\mathtt{b})} = (s_i, s_{(i+1)}, ...s_{(i+w-1)})\right\}}.$$

Equivalently, the resulting score can be expressed by a position weight matrix with weights $w_{k\alpha} = \log q_{k\alpha}^{(\mathtt{p})} - \log q_{k\alpha}^{(\mathtt{b})}$, so that

$$z_{\mathtt{m}i} = \sum_{k=1}^{w} \sum_{\alpha \in \{\mathtt{a},\mathtt{c},\mathtt{g},\mathtt{t}\}} w_{k\alpha} \cdot [s_{i+k-1} = \alpha].$$

5.2.9 Score integration

In the score integration step, for each possible window i, the footprint and the motif scores $z_{\mathtt{f}i}$ and $z_{\mathtt{m}i}$ are integrated into a footif score z_i. The most naïve score integration method would be the sum of the two scores. However, this suffers from two problems. First, the footprint scores typically follow a completely different distribution and have a variance several orders of magnitudes larger than the motif scores; a simple sum would result in the footprint score completely dominating the

overall score. Second, even if both scores were in a similar range, it cannot be assumed that the counts and the sequence have the same importance; the relative importance of each set of features should be learned as well.

The first issue is addressed by rescaling the two score distributions to a gaussian distribution with mean 0 and variance 1. This is accomplished using ranks: given a vector $\boldsymbol{x}$ of length n, we denote by $\text{rank}(\boldsymbol{x})_i$ the position of element x_i after that vector $\boldsymbol{x}$ is sorted by decreasing values, so that the highest element of $\boldsymbol{x}$ has rank 1 and the lowest has rank n. Moreover, we use the inverse cumulative distribution function of the gaussian distribution F^{-1}, where F is defined by:

$$F(x) = \int_{-\infty}^{x} (2\pi)^{-0.5} \exp(-0.5x^2).$$

Denoting with n the total number of windows $|I|$, and therefore the length of vectors $\boldsymbol{z}_{\mathtt{f}}$ and $\boldsymbol{z}_{\mathtt{m}}$, the rescaled scores are:

$$\begin{aligned} \dot{z}_{\mathtt{f}i} &= F^{-1}(1 - \text{rank}(\boldsymbol{z}_{\mathtt{f}})_i/(n+1)), \\ \dot{z}_{\mathtt{m}i} &= F^{-1}(1 - \text{rank}(\boldsymbol{z}_{\mathtt{m}})_i/(n+1)). \end{aligned}$$

The second issue is accounted for by two coefficients $\beta_{\mathtt{f}}$ and $\beta_{\mathtt{m}}$, such that

$$z_i = \beta_{\mathtt{m}} \dot{z}_{\mathtt{m}i} + \beta_{\mathtt{f}} \dot{z}_{\mathtt{f}i}.$$

More precisely, using linear discriminant analysis, a linear footif classifier is trained by using the rescaled scores $\dot{z}_{\mathtt{m}i}$ and $\dot{z}_{\mathtt{f}i}$ as features and the classification from the previous iteration as training data. Let $I^{(\mathtt{p})}$ and $I^{(\mathtt{b})}$ denote the set of windows classified as peaks or background from the previous iteration and $|I^{(\mathtt{p})}|$ and $|I^{(\mathtt{b})}|$ the number of bound and unbound windows, respectively. The weights $\beta_{\mathtt{m}}$ and $\beta_{\mathtt{f}}$ are computed using the following vectorial formula:

$$\begin{pmatrix} \beta_{\mathtt{m}} \\ \beta_{\mathtt{f}} \end{pmatrix} = \begin{pmatrix} \delta_{\mathtt{m}} \\ \delta_{\mathtt{f}} \end{pmatrix} \begin{bmatrix} \sigma_{\mathtt{mm}} & \sigma_{\mathtt{mf}} \\ \sigma_{\mathtt{fm}} & \sigma_{\mathtt{ff}} \end{bmatrix}^{-1}.$$

Where $\boldsymbol{\delta}$ and $\boldsymbol{\sigma}$ are given by:

$$\delta_u = |I^{(\mathtt{p})}|^{-1} \sum_{i \in I^{(\mathtt{p})}} \dot{z}_{ui} - |I^{(\mathtt{b})}|^{-1} \sum_{i \in I^{(\mathtt{b})}} \dot{z}_{ui} \qquad u \in \{\mathtt{m}, \mathtt{f}\},$$

$$\sigma_{uv} = \sum_{j \in \{\mathtt{p},\mathtt{b}\}} \left\{ |I^{(j)}|^{-1} \sum_{i \in I^{(j)}} \dot{z}_{ui} \dot{z}_{vi} - |I^{(j)}|^{-2} \sum_{i \in I^{(j)}} \dot{z}_{ui} \sum_{i \in I^{(j)}} \dot{z}_{vi} \right\} \qquad u, v \in \{\mathtt{m}, \mathtt{f}\}.$$

For more information about linear discriminant analysis see, for instance, McLachlan [124].

5.2.10 Thresholding

In the thresholding step putative binding sites are obtained from the footif score. This is done by selecting the n_t highest scoring, non overlapping windows as a set of putative binding sites, where n_t is a parameter of the algorithm. Overlaps are avoided by selecting windows by decreasing score values, and by excluding at each step all windows overlapping the selected one. After n_t peaks have been determined, the classification of the windows in I as peaks ($I^{(\mathtt{p})}$) or background ($I^{(\mathtt{b})}$) is updated and used for the next iteration of the algorithm (as shown in Figure 5.3).

5.3 Results

We set out to test whether our approach is able to recover biologically meaningful footprints and motifs compatible with those annotated in the JASPAR database [49]. To this end we ran it on 4 published ChIP-exo experiments on human cells:

- FOXA1: ChIP-exo experiment for the transcription factor FOXA1 on MCF7 cells (from Serandour *et al.* [122]),
- GR: ChIP-exo experiment for the glucocorticoid receptor on IMR90 cells (from Starick *et al.* [125]),
- CTCF: ChIP-exo experiment for the transcription factor CTCF on HeLa cells (from Rhee & Pugh [41]),
- ER: ChIP-exo experiment for the estrogen receptor α on MCF7 cells (from Serandour *et al.* [122]).

The algorithm was run on a subset of the genome enriched for ChIP-exo reads of approximately 15 million bases. The window size w was set to 100 and the number of putative binding sites chosen after thresholding n_t was chosen roughly equal to 0.1% of the total number of binding sites. Each footif is displayed by a sequence logo (see Subsection 2.2.1) aligned to an average peak shape. These are derived by averaging the count and the sequence data across the sites recognized by the footif. The figures in the remainder of this subsection focus on the most representative footifs and display only the most information-rich portion of the 100 base pairs window. See Figure A.8 for a more comprehensive overview of the obtained results.

Figure 5.5 shows the first footprint obtained in the FOXA1 dataset. The recovered motif matches very well with the one reported in JASPAR (see Figure A.9). The average peak shape exhibits a sharp peak on the antisense strand, suggesting that the crosslinks block exonuclease digestion very precisely. In contrast, when the digestion progresses on the sense strand from left to right, the border of the protected DNA region seems to be less defined, which can be explained by multiple weak crosslinks (see Figure 5.6). This interpretation has been put forward also in Starick *et al.* [125], to which we refer for a structural justification. Note how the obtained average peak shape resembles that shown in Figure 5.2. The procedure

that led to the two results are, however, very different. In Figure 5.2 the putative binding sites were obtained by assuming that FOXA1 recognizes a known motif. In Footifind, by contrast, no assumptions are made about the sequence preferences of the TF, which are automatically detected together with the average peak shape.

The second recovered footif (see Figure A.8) is almost identical to the first, except that it is oriented in the opposite direction. This implies that the motif and the footprint in the second footif are reverse-complemented versions of those in the first. Note that the reverse-complement of an average peak shape is obtained by reversing and swapping the two curves corresponding to sense and anti-sense strand.

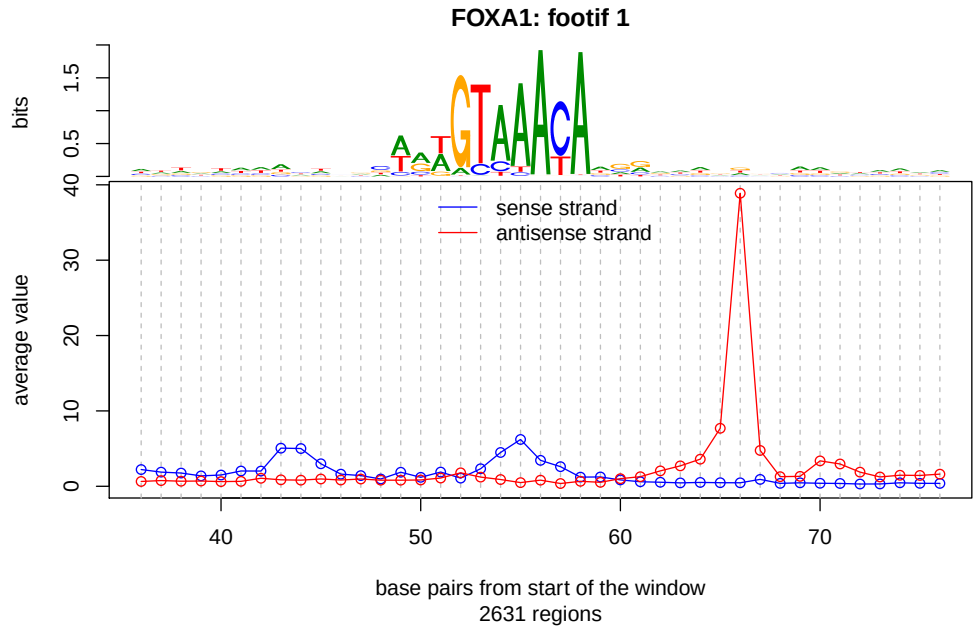

Figure 5.5: *First footif returned by Footifind in the FOXA1 dataset.*

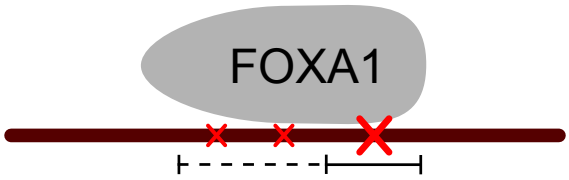

Figure 5.6: *Interpretation of the footif of Figure 5.5. The red crosses represent the crosslinks and the horizontal bar represents protected (solid) and semi-protected (dashed) DNA. The footprint suggests that the right border is protected very precisely by a strong crosslink, while the left border is not very well defined and might originate from weak crosslinks.*

Figure 5.7 shows the first footprint obtained in the GR dataset. As in the previous case, the detected motif matches the one reported in JASPAR (see Figure

A.9). Both the footprint and the motif seem to be palindromic (i.e identical to their reverse-complements) and the average peak shape shows two main protection sites, delimited by two peak pairs. This is in agreement with the fact that GR tends to bind to the DNA as a homodimer [126]. Each of the two peak pairs, therefore, can be explained as one bound GR unit protecting a distinct DNA region (see Figure 5.8).

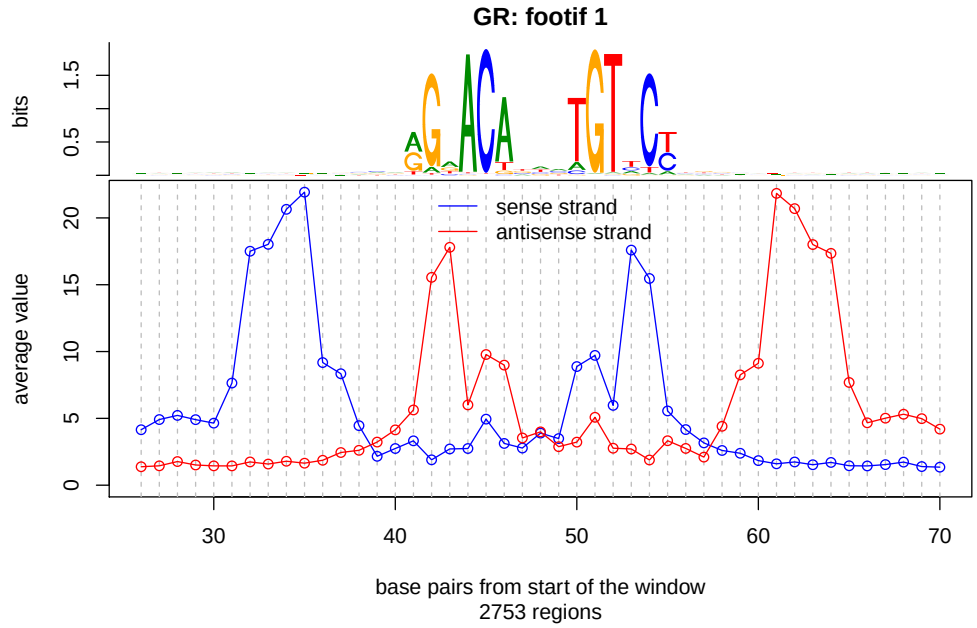

Figure 5.7: *First footif returned by Footifind in the GR dataset.*

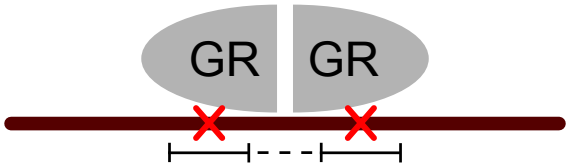

Figure 5.8: *Interpretation of the footif of Figure 5.7. The red crosses represent the crosslinks and the horizontal bar represents protected (solid) and semi-protected (dashed) DNA. The footprint suggests that each of the GR units is strongly crosslinked with the DNA. The region between the two units is not completely protected, causing the two internal peaks to be present, but smaller than the external ones.*

The second recovered footif is almost identical to the first (not shown). The third recovered footif, instead, looks very different from the first two and very similar to the FOXA1 footif. This has already been noted in Starick *et al.* [125], to which we refer for an in-depth discussion. The footif suggests that in the ChIP-exo experiment

for GR also a protein of the FOX family was immunoprecipitated. This might imply that GR binding depends, or is favored by the nearby binding of a FOX protein. FOX proteins, in fact, are known to be pioneer TFs, as they can bind condensed chromatin and cause the bound region to become accessible to other TFs [127]. Other explanations are possible. For instance, there could be unknown, biologically relevant interactions between the two proteins, or the sample might have been contaminated. This example also shows Footifind's potential: it not only locates binding events, but it can also discern distinct and biologically relevant groups of events (the footifs), so that each binding event can be annotated with the group it belongs to.

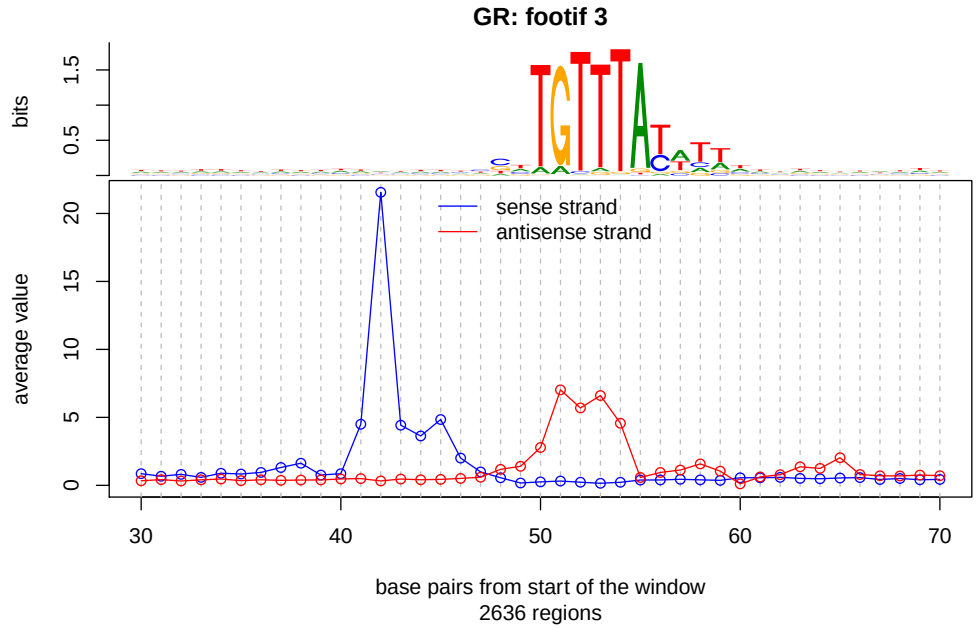

Figure 5.9: *Third footif returned by Footifind in the GR dataset.*

The first recovered footif in the CTCF dataset is shown in Figure 5.10. Again, the recovered motif matches very well the one reported in JASPAR (see Figure A.9). The average peak shape shows that the borders of the protected DNA region are relatively distant from one another and not very well defined. The CTCF protein is known to have multiple DNA-binding domains (11 zinc fingers), which can be used in different combinations to recognize different sequences and perform different regulatory functions [128]. The poorly defined footprint of Figure 5.10, therefore, could be the consequence of an heterogeneous 3D structure of the DNA-protein complex (see Figure 5.11). Similarly as in the FOXA1 dataset, the second footif is almost identical to the first, but with opposite orientation.

In the ER dataset, the first footif (shown in Figure A.8) does not seem biologically relevant. The motif strongly matches the Alu repeat sequence and the footprint does not exhibit clear peak pairs or a strong positional dependence. The second footif, by contrast, shown in Figure 5.12, is most likely genuine, as the motif agrees with the

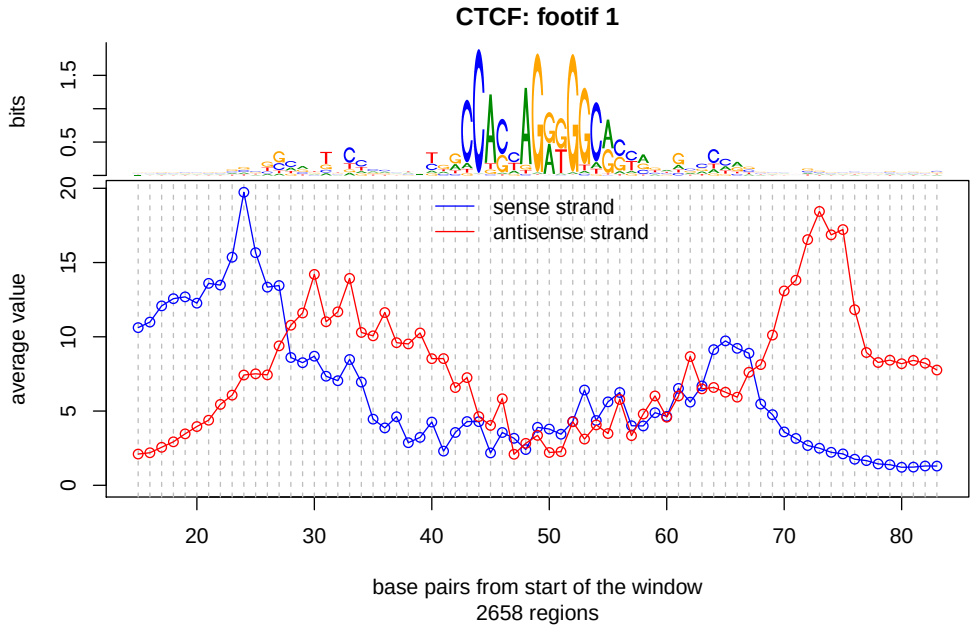

Figure 5.10: *First footif returned by Footifind in the CTCF dataset.*

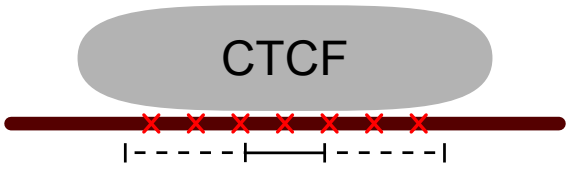

Figure 5.11: *Interpretation of the footif of Figure 5.10. The red crosses represent the crosslinks and the horizontal bar represents protected (solid) and semi-protected (dashed) DNA. The footprint suggests that CTCF forms many weak and variable crosslinks with the DNA along a relatively large region, resulting in weakly defined borders.*

one reported in JASPAR (see Figure A.9) and the footprint seems plausible, even though very noisy. As for GR, ER is a nuclear receptor known to bind the DNA as a homodimer [126], which explains why the motif and the footprint are almost perfect palindromes. However it is difficult to interpret the average peak shape in terms of crosslinks. It is unclear whether the low signal-to-noise ratio is due to the inaccuracy of the algorithm or to a poor quality of the experiment.

To summarize, the algorithm recovered the motif known from the literature in all 4 datasets without any prior information. Moreover, the obtained footprints are in agreement with what is known about the 3D structure of the protein-DNA binding and can be used to generate new hypotheses. Strikingly, in the GR dataset two qualitatively different footifs were obtained, showing that from a single ChIP-exo experiment different families of peaks can be detected.

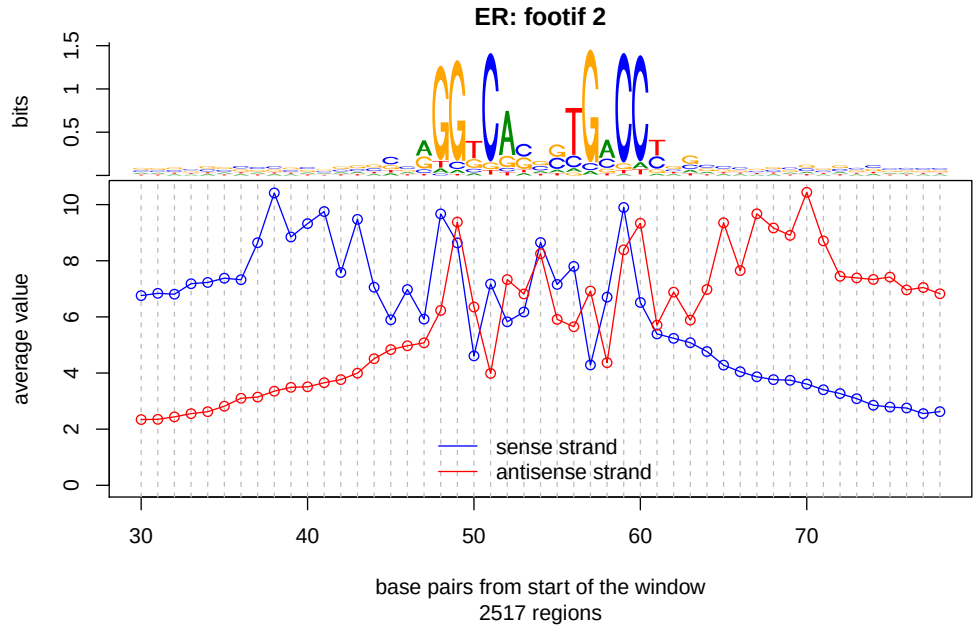

Figure 5.12: *Second footif returned by Footifind in the ER dataset.*

5.4 Discussion

We presented Footifind: an algorithm that derives an integrated understanding of protein binding from ChIP-exo data, for which we coined the term footif. A footif comprises two strictly coupled models for two different data sources: a model of the DNA sequences that the TF binds to (the motif), and a model of how the ChIP-exo reads tend to be distributed in proximity of the binding sites (the footprint). Footifind has several important practical applications. First, by using sequence and count data simultaneously, our method bases its predictions on multiple pieces of evidence and therefore enhances traditional peak calling approaches for the definition

of putative binding sites. Second, different categories of binding events can be detected, which can have unexpected biological implications. Third, in the analyzed datasets our method correctly identified the expected motifs, suggesting that it might enhance de-novo motif finding algorithms by accurately modeling the read abundances. Lastly, the obtained footprints can be used to interpret the structure of the protein-DNA binding and to formulate new hypotheses.

However, Footifind also has some limitations, which will be addressed by future research. First, there is no function being optimized in the iterative approach. Even though it makes intuitive sense that each loop improves the description of the motif, footprint and candidate binding sites, this is not guaranteed by any objective measurement. As a consequence, there is no objective quality measure for the returned footifs. Second, the thresholding mechanism is very rigid, because the number of putative binding site n_t is not learned from the data, but is a fixed parameter. Setting n_t too low compared to the biological truth could result in many similar footifs. If n_t is set too high, the method might fail or return footifs with a low signal-to-noise ratio. Despite these limitations, this algorithm shows a great potential in the analysis of ChIP-exo data, and it might also be useful for the analysis of DNAse-seq and ATAC-seq data.

Chapter 6

Conclusion

We presented 3 novel algorithms for automatically detecting important patterns in ChIP-seq count data. By measuring protein-DNA interactions genome-wide, this protocol allowed us to analyze poorly understood biological processes affecting gene expression: nucleosome positioning, chromatin states, and transcription factor binding.

Although diverse in their biological motivation, the computational problems that we addressed were similar. First, the input of our algorithms are count signals: sequences of read counts quantifying the presence of a protein along the genome. Second, different biological phenomena were described as a patterns recurring in one or more count signals. Third, we developed algorithms to learn these patterns and to locate their occurrences efficiently.

Moreover, NucHunter, EpiCSeg and Footifind should not be viewed as independent projects, but rather as a continuous path in the exploration of count patterns. To illustrate this we will distinguish positional patterns, describing the counts at adjacent positions of the same signal, and vertical patterns, describing the counts at the same position across multiple signals.

NucHunter detects well-positioned nucleosomes assuming that the counts at nucleosome positions follow a positional pattern. After preprocessing and integrating multiple experiments, in fact, a linear filter with a particular impulse response is used to locate nucleosomes in the consensus signal. This algorithm proved to be not only more accurate, but also considerably faster than previous approaches. However, many parameters were determined based on expert knowledge rather than inferred from the data. In fact, although the average fragment length could be estimated from the cross-correlation function, the filter was chosen based on our expectations about the count pattern at nucleosome positions. A more principled approach was prevented by the difficulty in defining nucleosome formation sites; nucleosomes, in fact, can form at different, incompatible locations in different cells or at different times, so that it is not clear how broad a peak should be for it to represent a single nucleosome.

In the footprint detection problem, by contrast, because of the sequence preferences of a TF and the high resolution of the ChIP-exo technique, TF binding sites could be defined more accurately. Footifind, therefore, should be seen as a

refinement of NucHunter, because the parameters used to detect a positional pattern are justified by an inference algorithm. Besides learning the count pattern (the footprint), Footifind also learns the motif recognized by the TF, thus providing an integrated model for the read counts and the sequence at TF binding sites. We showed that this approach, based on the innovative concept of a footif, was able to recover the correct motifs and interpretable footprints for the 4 different TFs that we analyzed.

Vertical patterns have been first explored in Chapter 3, after that nucleosomes were predicted from multiple ChIP-seq experiments. The combination of histone marks active on nucleosomes were attributed to a small number of biologically relevant patterns called chromatin states. These patterns, inferred using a standard clustering algorithm, were shown to be highly related to the transcription cycle and to the regulatory processes taking place on the chromatin. However, arbitrary decisions had to be made in transforming the read counts and selecting the clustering method. Moreover, the resulting chromatin state annotation was limited to the occurrences of well-positioned nucleosomes.

In the chromatin segmentation problem, by contrast, the concept of chromatin state was decoupled from that of positioned nucleosome and the read counts were directly used as input. EpiCSeg, therefore, should be seen as a refinement of the nucleosome clustering because it annotates the entire genome and it formulates all assumptions in a probabilistic model for vertical count patterns. We compared EpiCSeg with a popular approach that transforms the input data arbitrarily; different quality metrics computed in different datasets suggested that EpiCSeg's annotation is more accurate and comprehensive.

Finally, we presented a unified probabilistic framework for positional and vertical count patterns: the combinations of histone marks characterizing a chromatin state and the peak shape characterizing a footprint were modeled with a negative multinomial distribution. This choice was suggested by the correlation and the overdispersion observed in the patterns, and it was shown to give accurate biological results. Moreover, we showed how this distribution can be efficiently fitted to large datasets, which makes it a computationally convenient model.

Overall, we designed algorithms that compare favorably with previous approaches and we implemented software that can be used by researchers to study gene regulation. However, our approaches also share a weakness. It is difficult to test the algorithms' predictions, as the problems are ill-posed. What range of positions can a nucleosome occupy for it to be considered a well-positioned nucleosome? What exactly are chromatin states and why should the negative multinomial distribution be the most appropriate model? In how many cells should a locus be bound by a TF for the locus to be called a TF binding site? Unfortunately these questions do not have a clear answer. Future research might focus on a more rigorous treatment of the aforementioned problems so that the predictions are easy to test and the performance easy to evaluate.

Bibliography

1. Mammana, A., Vingron, M. & Chung, R. Inferring nucleosome positions with their histone mark annotation from ChIP data. *Bioinformatics* **29,** 2547–54 (Oct. 2013).
2. Mammana, A. & Chung, R. Chromatin segmentation based on a probabilistic model for read counts explains a large portion of the epigenome. *Genome Biol.* **16,** 151 (July 2015).
3. Mammana, A. & Helmuth, J. *bamsignals: Extract read count signals from bam files* `http://bioconductor.org/packages/release/bioc/html/bamsignals.html`. Web Page. 2015.
4. Knuth, D. E. *Computer Literacy Bookshops Interview with Donald Knuth* `http://tex.loria.fr/litte/knuth-interview`. Web Page. 1993.
5. Watson, J. D. & Crick, F. H. Molecular structure of nucleic acids: a structure for deoxyribose nucleic acid. *Nature* **248,** 765 (Apr. 1974).
6. Lander, E. S. *et al.* Initial sequencing and analysis of the human genome. *Nature* **409,** 860–921 (Feb. 2001).
7. Venter, J. C. *et al.* The sequence of the human genome. *Science* **291,** 1304–51 (Feb. 2001).
8. Crick, F. Central dogma of molecular biology. *Nature* **227,** 561–3 (Aug. 1970).
9. Flynt, A. S. & Lai, E. C. Biological principles of microRNA-mediated regulation: shared themes amid diversity. *Nat. Rev. Genet.* **9,** 831–42 (Nov. 2008).
10. Mercer, T. R. & Mattick, J. S. Structure and function of long noncoding RNAs in epigenetic regulation. *Nat. Struct. Mol. Biol.* **20,** 300–7 (Mar. 2013).
11. Garneau, N. L., Wilusz, J. & Wilusz, C. J. The highways and byways of mRNA decay. *Nat. Rev. Mol. Cell Biol.* **8,** 113–26 (Feb. 2007).
12. Wang, C., Peterson, S. E. & Loring, J. F. Protein post-translational modifications and regulation of pluripotency in human stem cells. *Cell Res.* **24,** 143–60 (Feb. 2014).
13. Petrascheck, M. *et al.* DNA looping induced by a transcriptional enhancer in vivo. *Nucleic Acids Res.* **33,** 3743–50 (2005).
14. Rahmann, S., Müller, T. & Vingron, M. On the power of profiles for transcription factor binding site detection. *Stat. Appl. Genet. Mol. Biol.* **2,** Article 7 (2003).

15. Roider, H. G., Kanhere, A., Manke, T. & Vingron, M. Predicting transcription factor affinities to DNA from a biophysical model. *Bioinformatics* **23,** 134–41 (Jan. 2007).

16. Zhang, Y. *et al.* Model-based analysis of ChIP-Seq (MACS). *Genome Biol.* **9,** R137 (2008).

17. Ji, H. *et al.* An integrated software system for analyzing ChIP-chip and ChIP-seq data. *Nat. Biotechnol.* **26,** 1293–300 (Nov. 2008).

18. Bock, C. & Lengauer, T. Computational epigenetics. *Bioinformatics* **24,** 1–10 (Jan. 2008).

19. Heard, E. & Martienssen, R. A. Transgenerational epigenetic inheritance: myths and mechanisms. *Cell* **157,** 95–109 (Mar. 2014).

20. Bird, A. DNA methylation patterns and epigenetic memory. *Genes Dev.* **16,** 6–21 (Jan. 2002).

21. Lodish, H. *et al. Molecular Cell Biology* (W. H. Freeman, 2000).

22. Luger, K. & Richmond, T. J. DNA binding within the nucleosome core. *Curr. Opin. Struct. Biol.* **8,** 33–40 (Feb. 1998).

23. Han, M. & Grunstein, M. Nucleosome loss activates yeast downstream promoters in vivo. *Cell* **55,** 1137–45 (Dec. 1988).

24. Segal, E. *et al.* A genomic code for nucleosome positioning. *Nature* **442,** 772–8 (Aug. 2006).

25. Struhl, K. & Segal, E. Determinants of nucleosome positioning. *Nat. Struct. Mol. Biol.* **20,** 267–73 (Mar. 2013).

26. Gaffney, D. J. *et al.* Controls of nucleosome positioning in the human genome. *PLoS Genet.* **8,** e1003036 (2012).

27. Yen, K. *et al.* Genome-wide nucleosome specificity and directionality of chromatin remodelers. *Cell* **149,** 1461–73 (June 2012).

28. Strahl, B. D. & Allis, C. D. The language of covalent histone modifications. *Nature* **403,** 41–5 (Jan. 2000).

29. Li, B., Carey, M. & Workman, J. L. The role of chromatin during transcription. *Cell* **128,** 707–19 (Feb. 2007).

30. Zentner, G. E. & Henikoff, S. Regulation of nucleosome dynamics by histone modifications. *Nat. Struct. Mol. Biol.* **20,** 259–66 (Mar. 2013).

31. Barski, A. *et al.* High-resolution profiling of histone methylations in the human genome. *Cell* **129,** 823–37 (May 2007).

32. Bernstein, B. E. *et al.* Methylation of histone H3 Lys 4 in coding regions of active genes. *Proc. Natl. Acad. Sci. U.S.A.* **99,** 8695–700 (June 2002).

33. Wagner, E. J. & Carpenter, P. B. Understanding the language of Lys36 methylation at histone H3. *Nat. Rev. Mol. Cell Biol.* **13,** 115–26 (Feb. 2012).

34. De Almeida, S. F. *et al.* Splicing enhances recruitment of methyltransferase HYPB/Setd2 and methylation of histone H3 Lys36. *Nat. Struct. Mol. Biol.* **18,** 977–83 (Sept. 2011).
35. Benevolenskaya, E. V. Histone H3K4 demethylases are essential in development and differentiation. *Biochem. Cell Biol.* **85,** 435–43 (Aug. 2007).
36. Creyghton, M. P. *et al.* Histone H3K27ac separates active from poised enhancers and predicts developmental state. *Proc. Natl. Acad. Sci. U.S.A.* **107,** 21931–6 (Dec. 2010).
37. Beisel, C. & Paro, R. Silencing chromatin: comparing modes and mechanisms. *Nat. Rev. Genet.* **12,** 123–35 (Feb. 2011).
38. Koch, C. M. *et al.* The landscape of histone modifications across 1% of the human genome in five human cell lines. *Genome Res.* **17,** 691–707 (June 2007).
39. Baker, M. Making sense of chromatin states. *Nat. Methods* **8,** 717–22 (Sept. 2011).
40. Johnson, D. S., Mortazavi, A., Myers, R. M. & Wold, B. Genome-wide mapping of in vivo protein-DNA interactions. *Science* **316,** 1497–502 (June 2007).
41. Rhee, H. S. & Pugh, B. F. Comprehensive genome-wide protein-DNA interactions detected at single-nucleotide resolution. *Cell* **147,** 1408–19 (Dec. 2011).
42. Song, L. & Crawford, G. E. DNase-seq: a high-resolution technique for mapping active gene regulatory elements across the genome from mammalian cells. *Cold Spring Harb Protoc* **2010,** pdb.prot5384 (Feb. 2010).
43. Wang, Z., Gerstein, M. & Snyder, M. RNA-Seq: a revolutionary tool for transcriptomics. *Nat. Rev. Genet.* **10,** 57–63 (Jan. 2009).
44. Feller, W. *An Introduction to Probability Theory and Its Applications* ISBN: 0471257087 (Wiley, Jan. 1968).
45. Stormo, G. D. DNA binding sites: representation and discovery. *Bioinformatics* **16,** 16–23 (Jan. 2000).
46. Liu, J. & Stormo, G. D. Context-dependent DNA recognition code for C2H2 zinc-finger transcription factors. *Bioinformatics* **24,** 1850–7 (Sept. 2008).
47. Zhao, Y., Ruan, S., Pandey, M. & Stormo, G. D. Improved models for transcription factor binding site identification using nonindependent interactions. *Genetics* **191,** 781–90 (July 2012).
48. Weirauch, M. T. *et al.* Evaluation of methods for modeling transcription factor sequence specificity. *Nat. Biotechnol.* **31,** 126–34 (Feb. 2013).
49. Sandelin, A. *et al.* JASPAR: an open-access database for eukaryotic transcription factor binding profiles. *Nucleic Acids Res.* **32,** D91–4 (Jan. 2004).
50. Anscombe, F. J. The statistical analysis of insect counts based on the negative binomial distribution. *Biometrics* **5,** 165–73 (June 1949).
51. Robinson, M. D. & Smyth, G. K. Moderated statistical tests for assessing differences in tag abundance. *Bioinformatics* **23,** 2881–7 (Nov. 2007).

52. Anders, S. & Huber, W. Differential expression analysis for sequence count data. *Genome Biol.* **11,** R106 (2010).

53. Love, M. I., Huber, W. & Anders, S. Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2. *Genome Biol.* **15,** 550 (2014).

54. Rashid, N. U. *et al.* ZINBA integrates local covariates with DNA-seq data to identify broad and narrow regions of enrichment, even within amplified genomic regions. *Genome Biol.* **12,** R67 (2011).

55. Pique-Regi, R. *et al.* Accurate inference of transcription factor binding from DNA sequence and chromatin accessibility data. *Genome Res.* **21,** 447–55 (Mar. 2011).

56. Casella, G. & Berger, R. L. *Statistical Inference* ISBN: 0534243126 (Duxbury Press, June 2001).

57. Sivia, D. S. *Data Analysis: a Bayesian Tutorial* ISBN: 0198568320 (Oxford University Press, July 2006).

58. Little, R. J. A. & Rubin, D. B. *Statistical analysis with missing data* ISBN: 0-471-80254-9 (John Wiley & Sons, Inc., 1986).

59. Reynolds, D. in *Encyclopedia of Biometrics* 659–663 (Springer, 2009).

60. Rabiner, L. A tutorial on hidden Markov models and selected applications in speech recognition. *Proc. IEEE* **77,** 257–286. ISSN: 0018-9219 (Feb. 1989).

61. Durbin, R., Eddy, S. R., Krogh, A. & Mitchison, G. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids* ISBN: 9780521540797 (Cambridge University Press, 2005).

62. Alberts, B. *et al. Molecular biology of the cell* 4th ed, 207–213. ISBN: 0815316208 (Garland Publishing, New York, 2002).

63. Bannister, A. J. & Kouzarides, T. Regulation of chromatin by histone modifications. *Cell Res.* **21,** 381–95 (Mar. 2011).

64. Jiang, C. & Pugh, B. F. Nucleosome positioning and gene regulation: advances through genomics. *Nat. Rev. Genet.* **10,** 161–72 (Mar. 2009).

65. Johnson, S. M. *et al.* Flexibility and constraint in the nucleosome core landscape of Caenorhabditis elegans chromatin. *Genome Res.* **16,** 1505–16 (Dec. 2006).

66. Schones, D. E. *et al.* Dynamic regulation of nucleosome positioning in the human genome. *Cell* **132,** 887–98 (Mar. 2008).

67. Mavrich, T. N. *et al.* Nucleosome organization in the Drosophila genome. *Nature* **453,** 358–62 (May 2008).

68. Mavrich, T. N. *et al.* A barrier nucleosome model for statistical positioning of nucleosomes throughout the yeast genome. *Genome Res.* **18,** 1073–83 (July 2008).

69. Bernstein, B. E. *et al.* The NIH Roadmap Epigenomics Mapping Consortium. *Nat. Biotechnol.* **28,** 1045–8 (Oct. 2010).

70. The ENCODE Project Consortium. The ENCODE (ENCyclopedia Of DNA Elements) Project. *Science* **306,** 636–640 (2004).

71. Landt, S. G. *et al.* ChIP-seq guidelines and practices of the ENCODE and modENCODE consortia. *Genome Res.* **22,** 1813–31 (Sept. 2012).

72. Song, Q. & Smith, A. D. Identifying dispersed epigenomic domains from ChIP-Seq data. *Bioinformatics* **27,** 870–1 (Mar. 2011).

73. Zang, C. *et al.* A clustering approach for identification of enriched domains from histone modification ChIP-Seq data. *Bioinformatics* **25,** 1952–8 (Aug. 2009).

74. Flores, O. & Orozco, M. nucleR: a package for non-parametric nucleosome positioning. *Bioinformatics* **27,** 2149–50 (Aug. 2011).

75. Zhang, Y. *et al.* Identifying positioned nucleosomes with epigenetic marks in human from ChIP-Seq. *BMC Genom.* **9,** 537 (2008).

76. Albert, I. *et al.* Translational and rotational settings of H2A.Z nucleosomes across the Saccharomyces cerevisiae genome. *Nature* **446,** 572–6 (Mar. 2007).

77. Weiner, A. *et al.* High-resolution nucleosome mapping reveals transcription-dependent promoter packaging. *Genome Res.* **20,** 90–100 (Jan. 2010).

78. Zhang, X. *et al.* Probabilistic inference for nucleosome positioning with MNase-based or sonicated short-read data. *PLoS ONE* **7,** e32095 (2012).

79. Becker, J., Yau, C., Hancock, J. M. & Holmes, C. C. NucleoFinder: a statistical approach for the detection of nucleosome positions. *Bioinformatics* **29,** 711–6 (Mar. 2013).

80. Schöpflin, R. *et al.* Modeling nucleosome position distributions from experimental nucleosome positioning maps. *Bioinformatics* **29,** 2380–6 (Oct. 2013).

81. Turner, B. M. The adjustable nucleosome: an epigenetic signaling module. *Trends Genet.* **28,** 436–44 (Sept. 2012).

82. Benjamini, Y. & Speed, T. P. Summarizing and correcting the GC content bias in high-throughput sequencing. *Nucleic Acids Res.* **40,** e72 (May 2012).

83. Deriche, R. in *Proceedings of the 2nd international conference on image processing* (1992), 263–267.

84. Van Vliet, L. J., Young, I. T. & Verbeek, P. W. in *Proceedings of the 14th international conference on pattern recognition* (1998), 509–514.

85. Hale, D. *Recursive Gaussian filters* tech. rep. CWP Report 546 (Center for Wave Phenomena, Colorado School of Mines, 2006).

86. Chung, R. & Vingron, M. Sequence-dependent nucleosome positioning. *J. Mol. Biol.* **386,** 1411–22 (Mar. 2009).

87. Weinberger, L. *et al.* Expression noise and acetylation profiles distinguish HDAC functions. *Mol. Cell* **47,** 193–202 (July 2012).

88. Brogaard, K., Xi, L., Wang, P. & Widom, J. A map of nucleosome positions in yeast at base-pair resolution. *Nature* **486,** 496–501 (June 2012).

89. Vavouri, T. & Lehner, B. Human genes with CpG island promoters have a distinct transcription-associated chromatin organization. *Genome Biol.* **13,** R110 (2012).

90. Margueron, R. & Reinberg, D. The Polycomb complex PRC2 and its mark in life. *Nature* **469,** 343–9 (Jan. 2011).

91. Adams, D. *et al.* BLUEPRINT to decode the epigenetic signature written in blood. *Nat. Biotechnol.* **30,** 224–6 (Mar. 2012).

92. Deutsches Epigenom Programm. *Welcome to DEEP* `http://www.deutsches-epigenom-programm.de/`. Web Page. 2012.

93. International Human Epigenome Consortium. *Welcome to IHEC* `http://www.ihec-epigenomes.org/`. Web Page. 2010.

94. Hon, G., Ren, B. & Wang, W. ChromaSig: a probabilistic approach to finding common chromatin signatures in the human genome. *PLoS Comput. Biol.* **4,** e1000201 (Oct. 2008).

95. Ernst, J. *et al.* Mapping and analysis of chromatin state dynamics in nine human cell types. *Nature* **473,** 43–9 (May 2011).

96. Filion, G. J. *et al.* Systematic protein location mapping reveals five principal chromatin types in Drosophila cells. *Cell* **143,** 212–24 (Oct. 2010).

97. Hoffman, M. M. *et al.* Unsupervised pattern discovery in human chromatin structure through genomic segmentation. *Nat. Methods* **9,** 473–6 (May 2012).

98. Won, K.-J. *et al.* Comparative annotation of functional regions in the human genome using epigenomic data. *Nucleic Acids Res.* **41,** 4423–32 (Apr. 2013).

99. Karlić, R. *et al.* Histone modification levels are predictive for gene expression. *Proc. Natl. Acad. Sci. U.S.A.* **107,** 2926–31 (Feb. 2010).

100. Levin, B. & Reeds, J. Compound multinomial likelihood functions are unimodal: Proof of a conjecture of IJ Good. *Ann. of Stat.* **5,** 79–87 (1977).

101. Brent, R. P. *Algorithms for minimization without derivatives* ISBN: 0-13-022335-2 (Prentice-Hall, 1973).

102. Robinson, M. D., McCarthy, D. J. & Smyth, G. K. edgeR: a Bioconductor package for differential expression analysis of digital gene expression data. *Bioinformatics* **26,** 139–40 (Jan. 2010).

103. Lun, A. T. L. & Smyth, G. K. De novo detection of differentially bound regions for ChIP-seq data using peaks and windows: controlling error rates correctly. *Nucleic Acids Res.* **42,** e95 (June 2014).

104. Shao, Z. *et al.* MAnorm: a robust model for quantitative comparison of ChIP-Seq data sets. *Genome Biol.* **13,** R16 (2012).

105. Xu, H., Wei, C.-L., Lin, F. & Sung, W.-K. An HMM approach to genome-wide identification of differential histone modification sites from ChIP-seq data. *Bioinformatics* **24,** 2344–9 (Oct. 2008).

106. Heinig, M. *et al.* histoneHMM: Differential analysis of histone modifications with broad genomic footprints. *BMC Bioinformatics* **16,** 60 (2015).

107. Dagum, L. & Menon, R. OpenMP: an industry standard API for shared-memory programming. *IEEE Comput. Science Eng.* **5,** 46–55 (Aug. 1998).

108. Eddelbuettel, D. *et al.* Rcpp: Seamless R and C++ integration. *J. of Stat. Softw.* **40,** 1–18 (2011).

109. Harrow, J. *et al.* GENCODE: producing a reference annotation for ENCODE. *Genome Biol.* **7 Suppl 1,** S4.1–9 (2006).

110. Burnham, K. P. & Anderson, D. R. *Model selection and multimodel inference: a practical information-theoretic approach* ISBN: 0387953647 (Springer Science & Business Media, 2002).

111. Magnani, L., Eeckhoute, J. & Lupien, M. Pioneer factors: directing transcriptional regulators within the chromatin environment. *Trends Genet.* **27,** 465–74 (Nov. 2011).

112. Neph, S. *et al.* An expansive human regulatory lexicon encoded in transcription factor footprints. *Nature* **489,** 83–90 (Sept. 2012).

113. Boyle, A. P. *et al.* High-resolution genome-wide in vivo footprinting of diverse transcription factors in human cells. *Genome Res.* **21,** 456–64 (Mar. 2011).

114. Li, L. GADEM: a genetic algorithm guided formation of spaced dyads coupled with an EM algorithm for motif discovery. *J. Comput. Biol.* **16,** 317–29 (Feb. 2009).

115. Quang, D. & Xie, X. EXTREME: an online EM algorithm for motif discovery. *Bioinformatics* **30,** 1667–73 (June 2014).

116. Narlikar, L. MuMoD: a Bayesian approach to detect multiple modes of protein-DNA binding from genome-wide ChIP data. *Nucleic Acids Res.* **41,** 21–32 (Jan. 2013).

117. Bailey, T. L. DREME: motif discovery in transcription factor ChIP-seq data. *Bioinformatics* **27,** 1653–9 (June 2011).

118. Machanick, P. & Bailey, T. L. MEME-ChIP: motif analysis of large DNA datasets. *Bioinformatics* **27,** 1696–7 (June 2011).

119. Luo, K. & Hartemink, A. J. Using DNase digestion data to accurately identify transcription factor binding sites. *Pac Symp Biocomput,* 80–91 (2013).

120. Sherwood, R. I. *et al.* Discovery of directional and nondirectional pioneer transcription factors by modeling DNase profile magnitude and shape. *Nat. Biotechnol.* **32,** 171–8 (Feb. 2014).

121. Guo, Y., Mahony, S. & Gifford, D. K. High resolution genome wide binding event finding and motif discovery reveals transcription factor spatial binding constraints. *PLoS Comput. Biol.* **8,** e1002638 (2012).

122. Serandour, A. A., Brown, G. D., Cohen, J. D. & Carroll, J. S. Development of an Illumina-based ChIP-exonuclease method provides insight into FoxA1-DNA binding properties. *Genome Biol.* **14,** R147 (2013).

123. Kumar, V. *et al.* Uniform, optimal signal processing of mapped deep-sequencing data. *Nat. Biotechnol.* **31,** 615–22 (July 2013).

124. McLachlan, G. *Discriminant analysis and statistical pattern recognition* (John Wiley & Sons, 2004).

125. Starick, S. R. *et al.* ChIP-exo signal associated with DNA-binding motifs provides insight into the genomic binding of the glucocorticoid receptor and cooperating transcription factors. *Genome Res.* **25,** 825–35 (June 2015).

126. Beato, M. Gene regulation by steroid hormones. *Cell* **56,** 335–44 (Feb. 1989).

127. Zaret, K. S. & Carroll, J. S. Pioneer transcription factors: establishing competence for gene expression. *Genes Dev.* **25,** 2227–41 (Nov. 2011).

128. Ohlsson, R., Renkawitz, R. & Lobanenkov, V. CTCF is a uniquely versatile transcription regulator linked to epigenetics and disease. *Trends Genet.* **17,** 520–7 (Sept. 2001).

129. Gentleman, R. C. *et al.* Bioconductor: open software development for computational biology and bioinformatics. *Genome Biol.* **5,** R80 (2004).

130. Liu, X., Mammana, A. & Bafna, V. Speeding up tandem mass spectral identification using indexes. *Bioinformatics* **28,** 1692–7 (July 2012).

Appendix A

Supplementary Figures

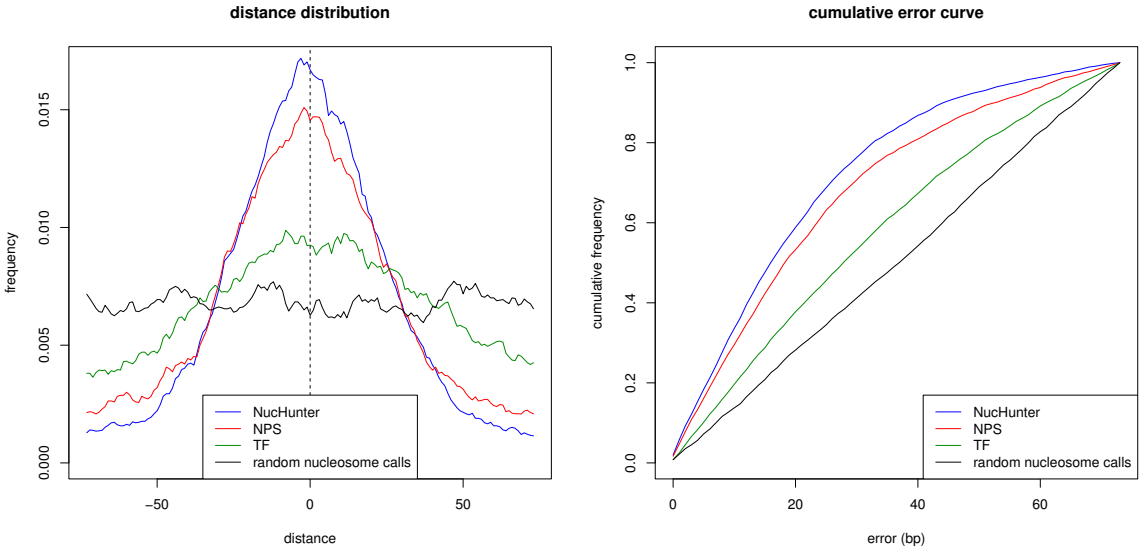

Figure A.1: *Distance distribution (left) and cumulative error curve (right). The top* 5000 *predictions from to the yeast dataset have been chosen from each tool. The distance distribution histogram has been smoothed with a running mean with a window size of* 10 *bps.*

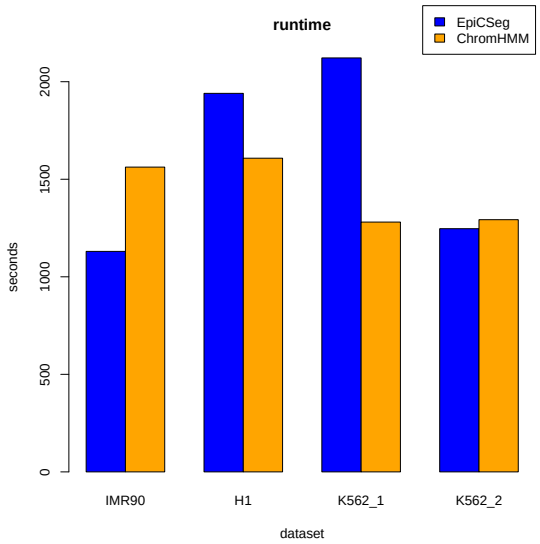

Figure A.2: *Runtime comparison between EpiCSeg and ChromHMM. The algorithms were run on a AMD Opteron computer with a clock speed of 2.66 GHz and using* 10 *computing threads.*

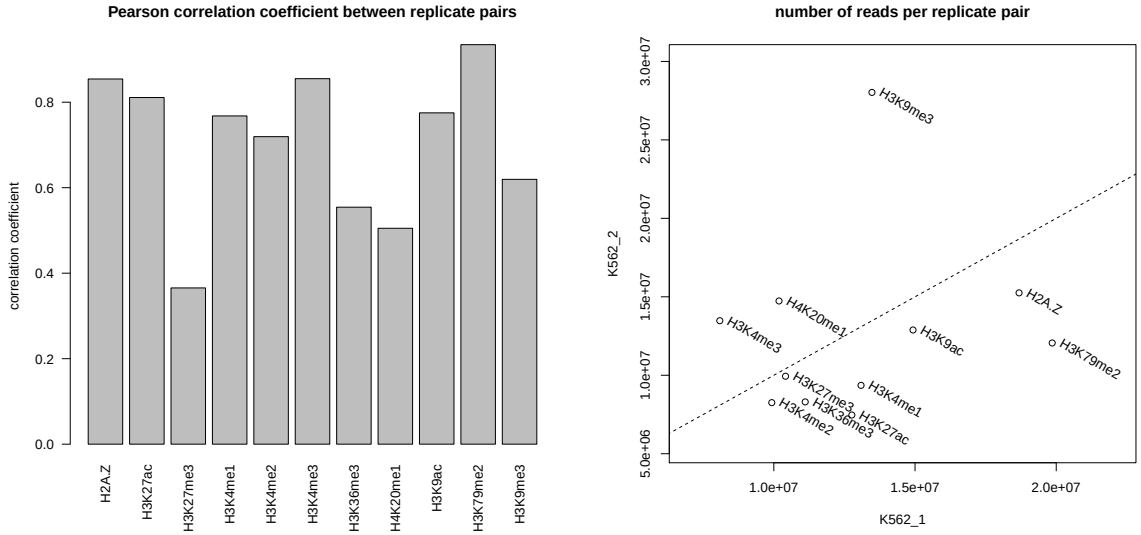

Figure A.3: *Similarity of each replicate pair. On the left, the Pearson correlation coefficient between replicate pairs. On the right, the total read count for each replicate pair. In both plots each replicate is transformed to a vector containing the read counts per bin.*

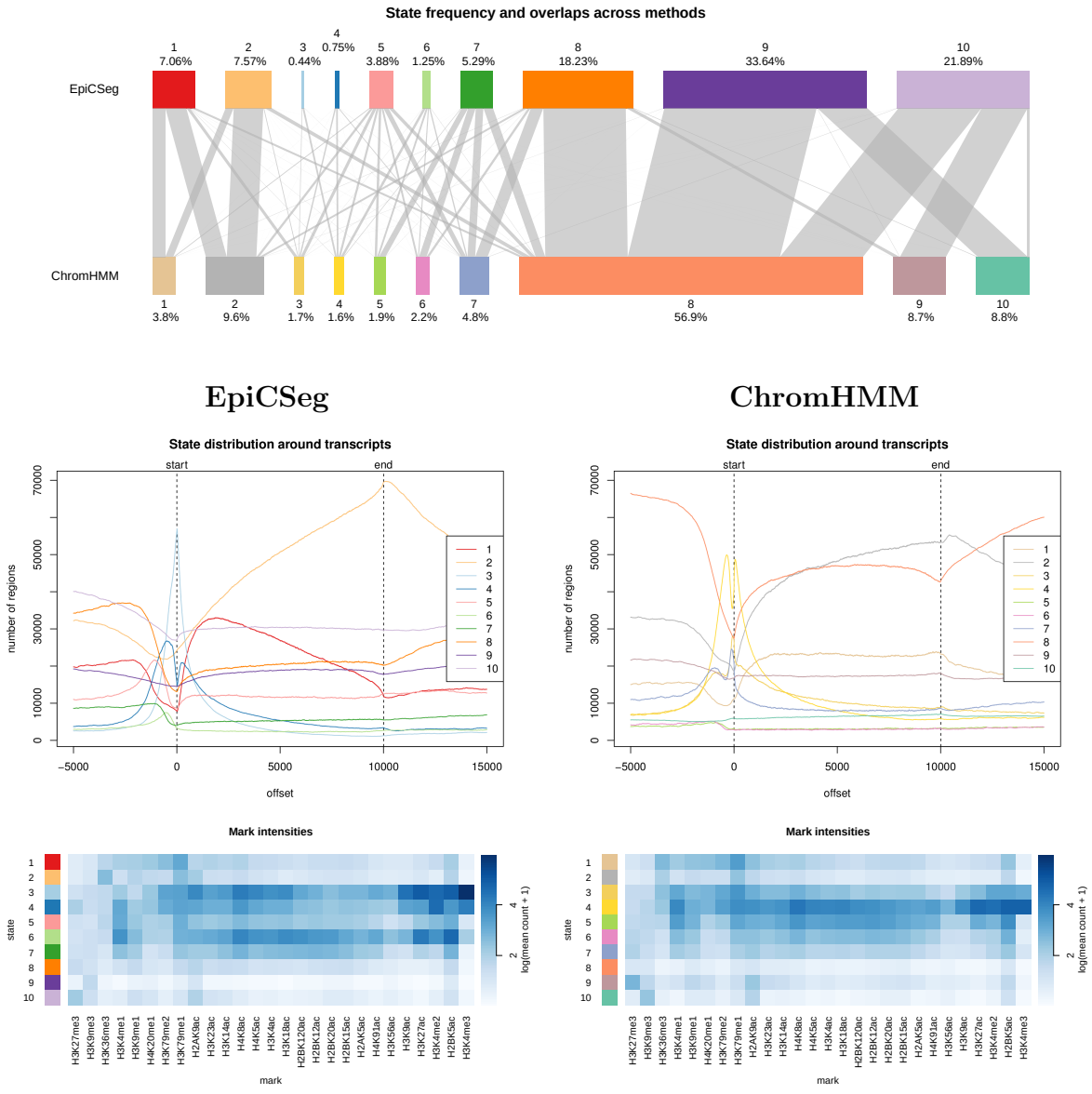

Figure A.4: *Genomic distribution of chromatin states in the IMR90 dataset. The choice of the colors is arbitrary.*

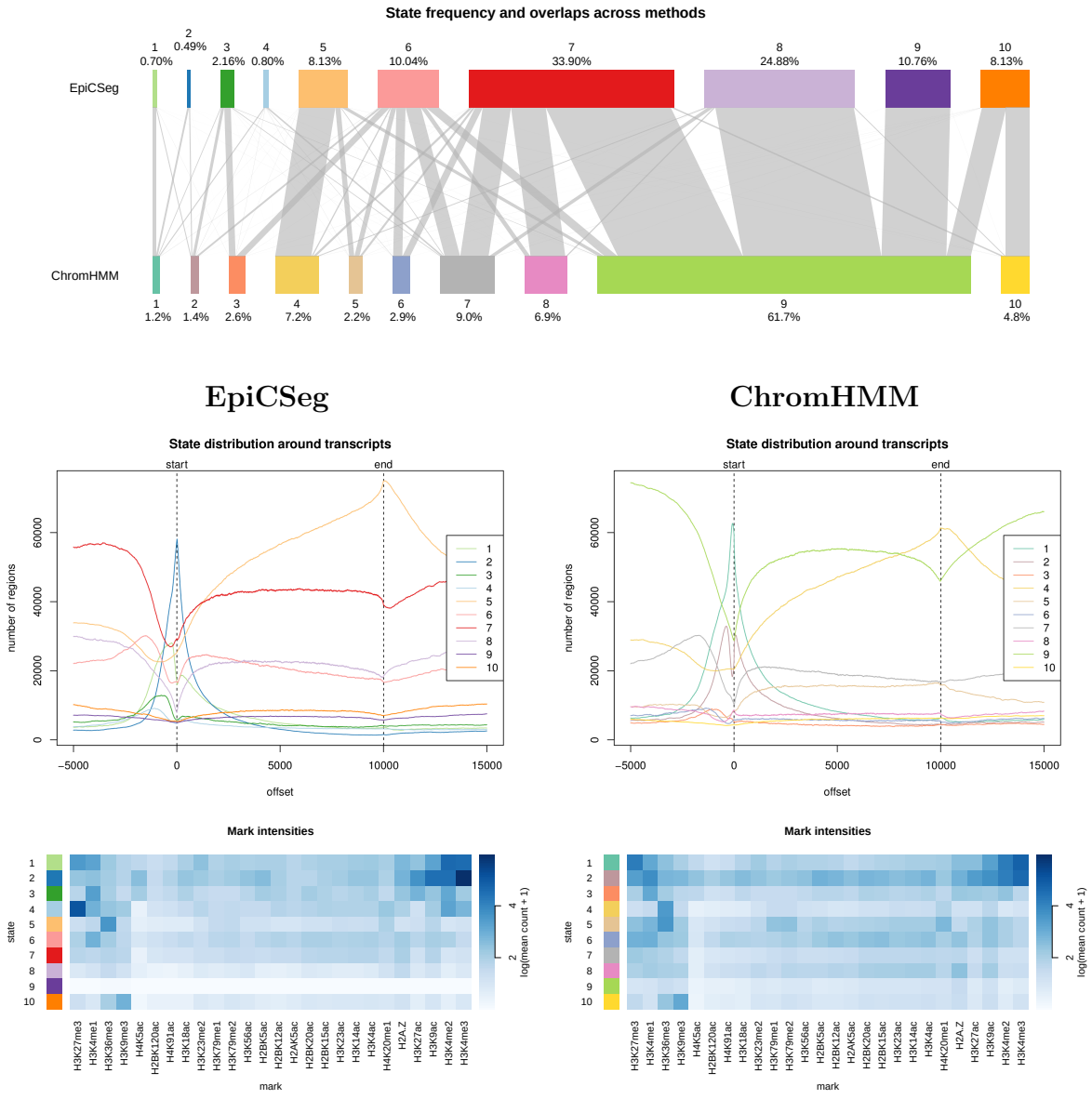

Figure A.5: *Genomic distribution of chromatin states in the H1 dataset. The choice of the colors is arbitrary.*

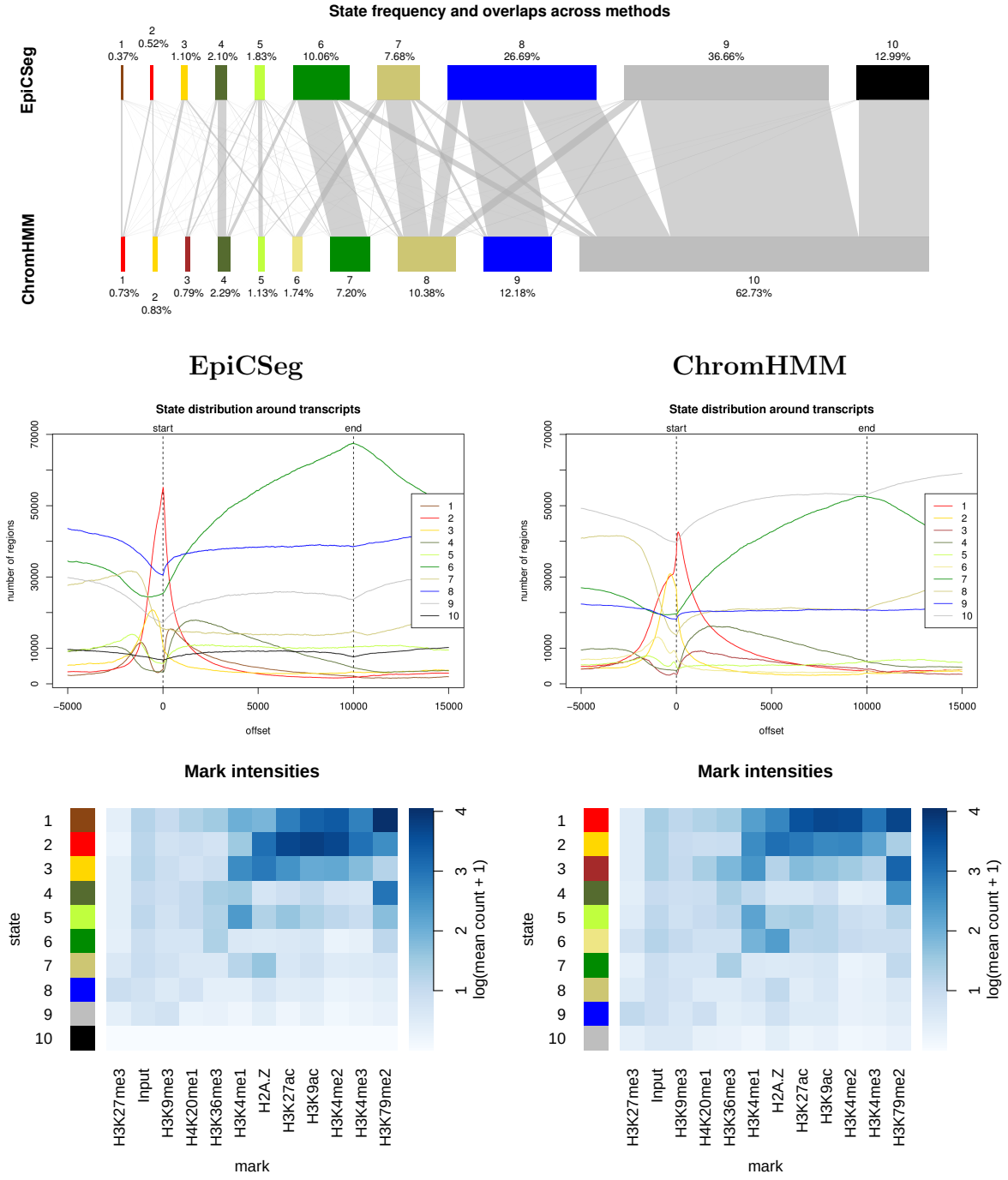

Figure A.6: *Genomic distribution of chromatin states in the K562_1 dataset. The choice of the colors is arbitrary.*

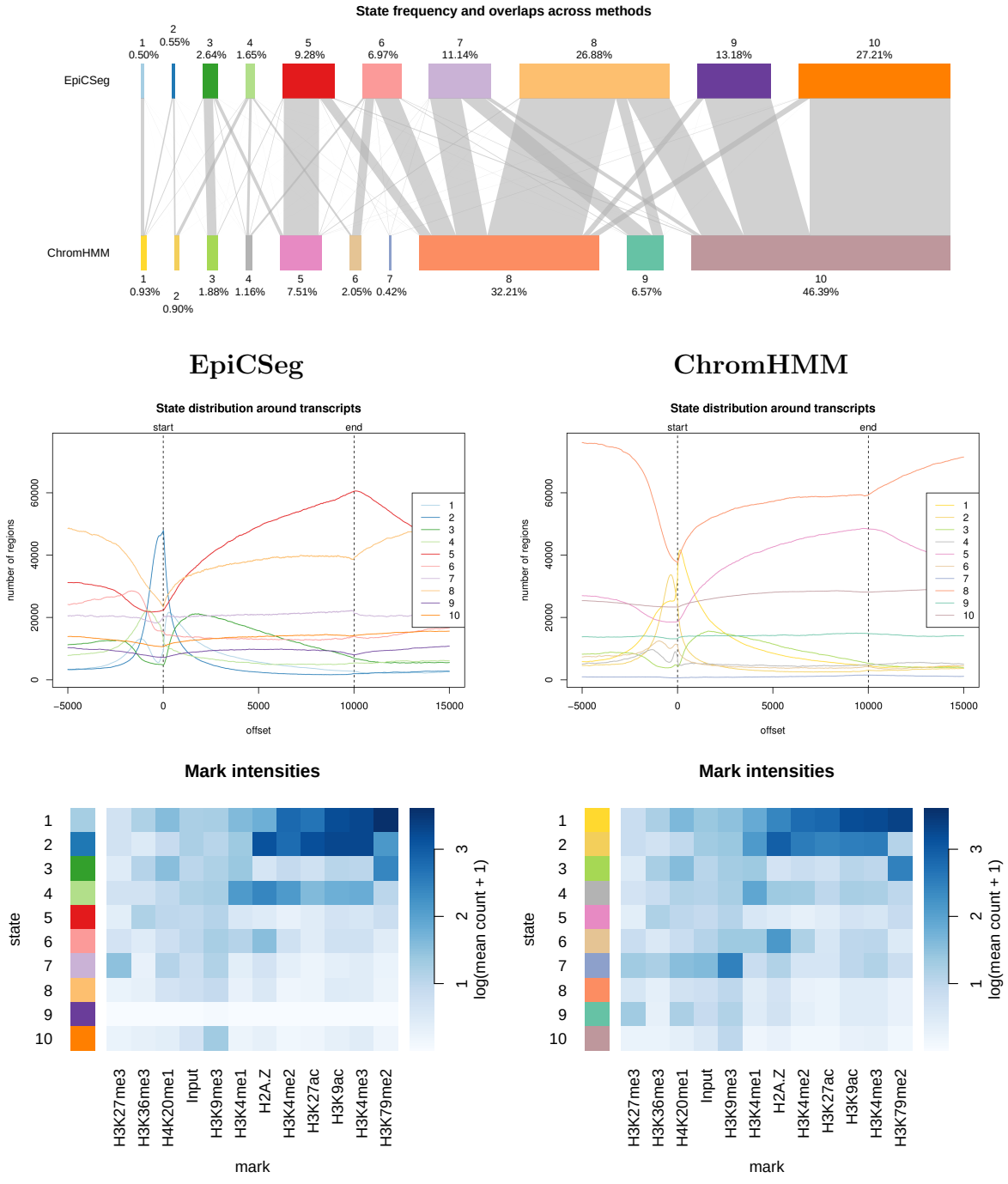

Figure A.7: *Genomic distribution of chromatin states in the K562_2 dataset. The choice of the colors is arbitrary.*

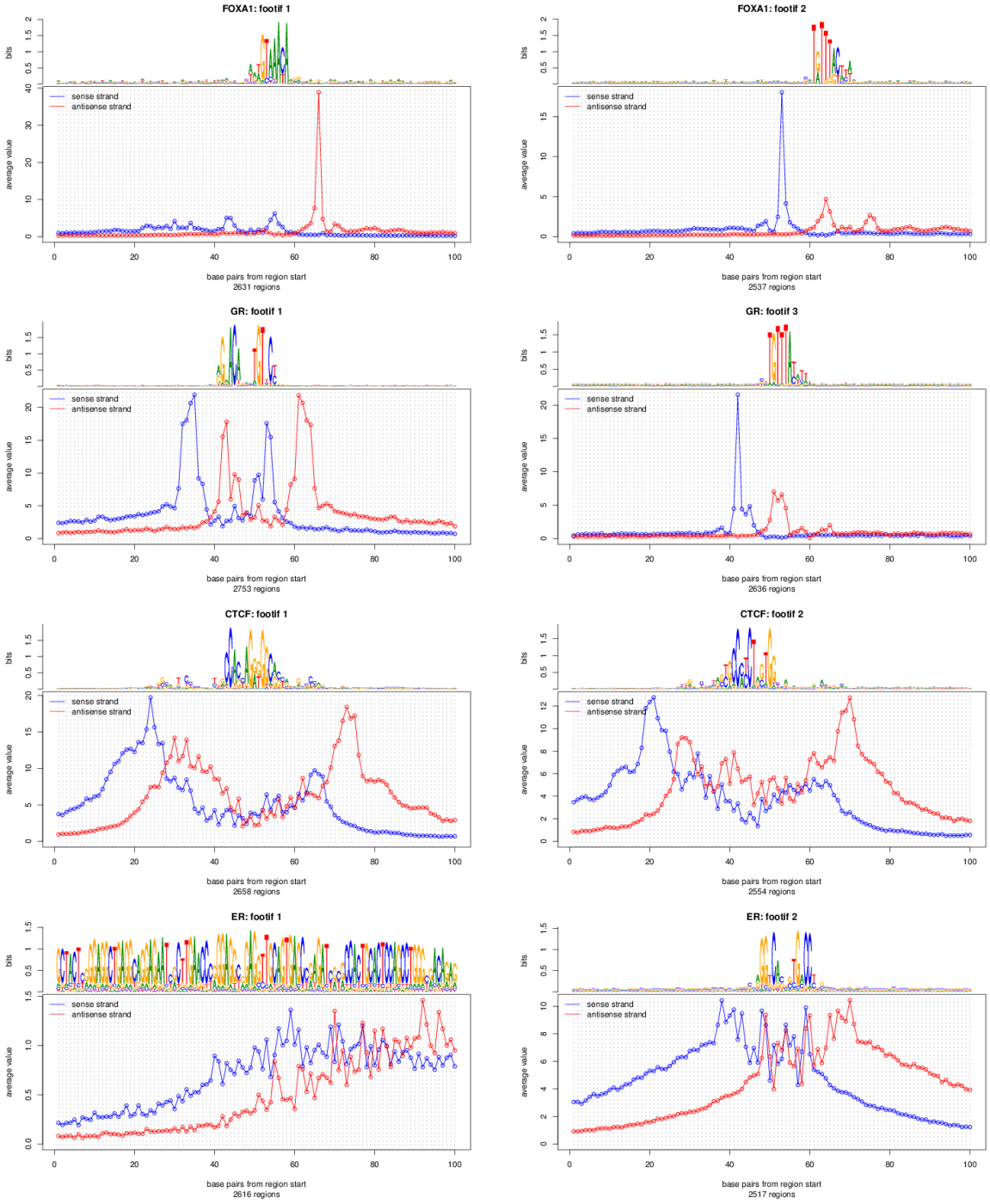

Figure A.8: *The footprint discovery algorithm on 4 different datasets. Each row represents a dataset, and each column a different footif. For each footif, the sequence and count averages across putative binding sites and for each position of the window are shown. The sequence average is displayed using the conventions for sequence logos, where the height of each letter shows how much the letter is overrepresented. These averages are also related to the parameters used in the motif and footprint classifiers.*

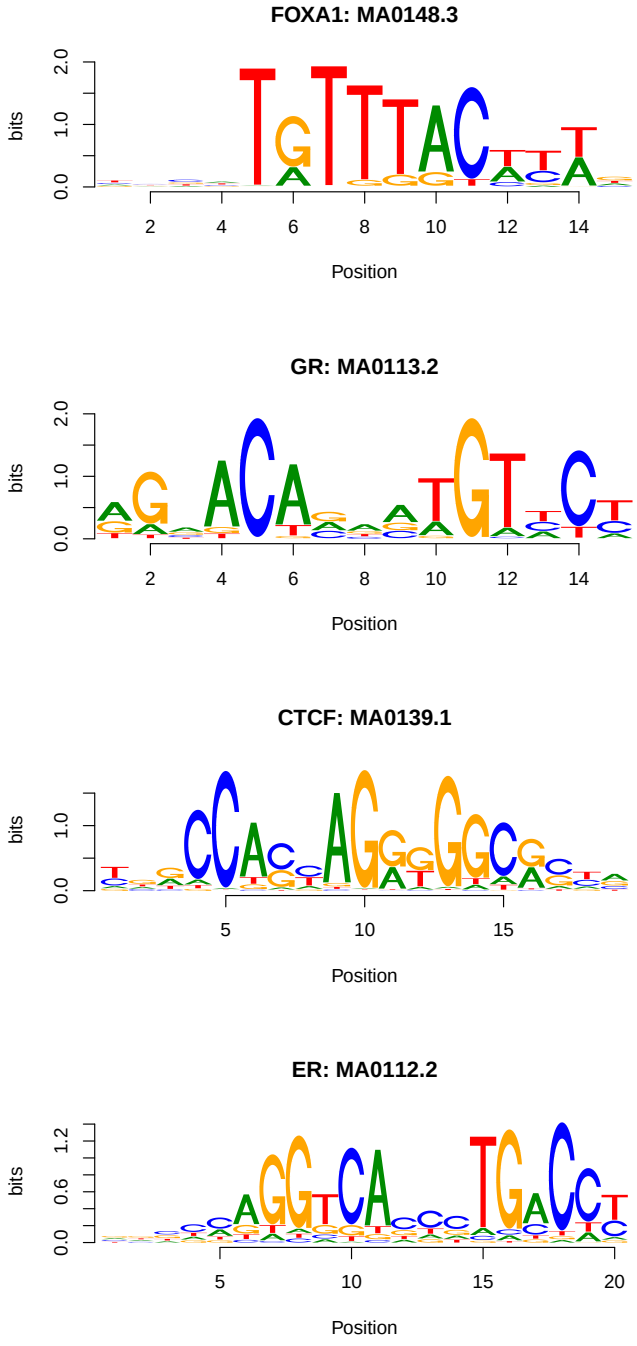

Figure A.9: *JASPAR motifs for the analyzed TFs. For each dataset, the most recent version of the motif was chosen (the ID is shown in the title of each plot). Note that some of the motifs are derived not only from human data, but also from data from other vertebrates, where very close protein homologues exist.*

Appendix B

Mathematical derivations

B.1 Optimization problems

In the following lemmas we denote by Δ^{m-1} the set of vectors of length m such that all vectors' elements are non-negative and their sum equals 1. In formulas:

$$\Delta^{m-1} = \{(t_1, t_2, ..., t_m) \in \mathbb{R}^m | \sum_{i=1}^{m} t_i = 1 \text{ and } t_i \geq 0 \text{ for all } i\}.$$

Lemma B.1.1. (Gibbs' inequality) *Let* $\boldsymbol{p} = (p_1, p_2, ..., p_m)$ *and* $\boldsymbol{q} = (q_1, q_2, ..., q_m)$ *denote two vectors in* Δ^{m-1}*. Then*

$$\sum_{i=1}^{m} p_i \log \frac{p_i}{q_i} \geq 0,$$

where, if $p_i = 0$*, we define* $p_i \log p_i/q_i$ *as* 0 *and if* $q_i = 0$ *and* $p_i > 0$*, we define* $p_i \log p_i/q_i$ *as* $+\infty$*. The equality holds if and only if* $p_i = q_i$ *for all* i*.*

Proof. Without loss of generality we can assume that for every i we have $q_i > 0$ or $p_i = 0$, otherwise the summation equals infinity and the strict inequality holds. Let I^+ denote the set of indices i such that p_i is strictly positive and I^0 the indices i such that p_i equals 0. We have

$$\sum_{i=1}^{m} p_i \log \frac{p_i}{q_i} = \sum_{i \in I^+} p_i \log \frac{p_i}{q_i} = -\sum_{i \in I^+} p_i \log \frac{q_i}{p_i}.$$

Because $\log x \leq x - 1$ for all $x > 0$, we have

$$-\sum_{i \in I^+} p_i \log \frac{q_i}{p_i} \geq -\sum_{i \in I^+} p_i \left(\frac{q_i}{p_i} - 1\right) = -\sum_{i \in I^+} q_i + \sum_{i \in I^+} p_i = \sum_{i \in I^0} q_i \geq 0.$$

For the equality to hold, first, $\log q_i/p_i$ must equal $q_i/p_i - 1$ for all i in I^+, second, q_i must be 0 for all i in I^0. Because $\log x$ equals $x - 1$ only for $x = 1$, these two conditions hold if and only if $p_i = q_i$ for all i. □

Lemma B.1.2. *Let $w_1, w_2, ...w_m$ denote m non-negative real numbers such that $\sum_{i=1}^m w_i > 0$. The unique solution to the optimization problem*

$$\arg\max_{\boldsymbol{p}\in\Delta^{m-1}} \sum_{i=1}^m w_i \log p_i$$

is

$$p_i = \frac{w_i}{\sum_{j=1}^m w_j} \text{ for } i = 1, 2, ...m.$$

Proof. Let $\hat{\boldsymbol{p}} = (\hat{p}_1, \hat{p}_n, ...\hat{p}_m)$ denote the vector such that $\hat{p}_i = w_i / \sum_{j=1}^m w_j$. We only need to show that $\sum_{i=1}^m w_i \log \hat{p}_i \geq \sum_{i=1}^m w_i \log q_i$ for all $\boldsymbol{q} \in \Delta^{m-1}$. This is an immediate consequence of Gibbs' inequality. In fact

$$\sum_{i=1}^m w_i \log \hat{p}_i - \sum_{i=1}^m w_i \log q_i = \sum_{i=1}^m w_i \log \frac{\hat{p}_i}{q_i} = \left(\sum_{i=1}^m w_i\right) \sum_{j=1}^m \hat{p}_j \log \frac{\hat{p}_j}{q_j} \geq 0.$$

□

In the following lemma we denote by $f_{\text{NB}}(c; \mu, r)$ the probability mass function of a negative binomial distribution with parameters $\mu \geq 0$ and $r > 0$ (see Subsection 2.2.2). In formulas:

$$f_{\text{NB}}(c; \mu, r) = \frac{\Gamma(r+c)}{\Gamma(r)c!} \left(\frac{\mu}{\mu+r}\right)^c \left(\frac{r}{\mu+r}\right)^r.$$

Lemma B.1.3. *Let $w_1, w_2, ...w_n$ denote n non-negative real numbers such that $\sum_{i=1}^n w_i > 0$ and let $c_1, c_2, ...c_n$ denote n non-negative integers. The unique solution to the optimization problem*

$$\arg\max_{\mu\geq 0} \sum_{i=1}^n w_i \log f_{NB}(c_i; \mu, r)$$

is

$$\mu = \frac{\sum_{i=1}^n w_i c_i}{\sum_{i=1}^n w_i}.$$

Proof. Let $o(\mu)$ denote the optimization function, which can be written as

$$o(\mu) = \sum_{i=1}^n w_i \log \frac{\Gamma(r+c_i)}{\Gamma(r)c_i!} + \sum_{i=1}^n w_i c_i \log \frac{\mu}{r+\mu} + \sum_{i=1}^n w_i r \log \frac{r}{r+\mu}.$$

Noting that the first term is independent of μ, the first derivative $o'(\mu)$ is:

$$o'(\mu) = \sum_{i=1}^n w_i c_i \frac{r}{\mu(r+\mu)} - \sum_{i=1}^n w_i \frac{r}{r+\mu} = \frac{r}{\mu(r+\mu)} \left(\sum_{i=1}^n w_i c_i - \mu \sum_{i=1}^n w_i\right).$$

Let $\hat{\mu}$ denote $\sum_{i=1}^n w_i c_i / \sum_{i=1}^n w_i$. In the special case where $\sum_{i=1}^n w_i c_i = 0$ the derivative is always negative, which implies that $\hat{\mu} = 0$ is a global maximum point. Otherwise, it is easy to see that $o'(\mu)$ is zero for $\mu = \hat{\mu}$, positive for $\mu < \hat{\mu}$ and negative for $\mu > \hat{\mu}$. This implies that $\hat{\mu}$ is the global maximum point of $o(\mu)$. □

B.2 KL divergence between negative multinomials

Let f_1 and f_2 denote the probability mass function of two negative multinomial distributions (see Subsection 2.2.3) with parameters $\mu_1, r, \boldsymbol{p}_1$ and $\mu_2, r, \boldsymbol{p}_2$, respectively:

$$f_j(\boldsymbol{x}) = \frac{\Gamma(r + x_+)}{\Gamma(r)} \left(\frac{\mu_j}{r + \mu_j}\right)^{x_+} \left(\frac{r}{r + \mu_j}\right)^{r} \prod_{l=1}^{m} \frac{p_{jl}^{x_l}}{x_l!},$$

where $\boldsymbol{x}$ is a vector of m non-negative integers, x_+ denotes $\sum_{l=1}^{m} x_l$ and j can be 1 or 2. Here we derive a formula for the Kullback-Leibler divergence of f_2 from f_1, defined as:

$$D_{\mathrm{KL}}(f_1||f_2) = \sum_{\boldsymbol{x} \in \mathbb{N}^m} f_1(\boldsymbol{x}) \log \frac{f_1(\boldsymbol{x})}{f_2(\boldsymbol{x})}.$$

The ratio between $f_1(\boldsymbol{x})$ and $f_2(\boldsymbol{x})$ is:

$$\frac{f_1(\boldsymbol{x})}{f_2(\boldsymbol{x})} = \left(\frac{r + \mu_2}{r + \mu_1} \frac{\mu_1}{\mu_2}\right)^{x_+} \left(\frac{r + \mu_2}{r + \mu_1}\right)^{r} \prod_{l=1}^{m} \left(\frac{p_{1l}}{p_{2l}}\right)^{x_l}.$$

Let now $\boldsymbol{X} = (X_1, X_2, ...X_m)$ denote a random vector whose distribution is specified by f_1, i.e. $\boldsymbol{X} \sim \mathrm{NM}(\mu_1, r, \boldsymbol{p}_1)$. The KL divergence can be written as the expectation of a function of $\boldsymbol{X}$:

$$\begin{aligned} D_{\mathrm{KL}}(f_1||f_2) =& \mathrm{E}\left[\log \frac{f_1(\boldsymbol{X})}{f_2(\boldsymbol{X})}\right] \\ =& \mathrm{E}\left[X_+ \log\left(\frac{r + \mu_2}{r + \mu_1} \frac{\mu_1}{\mu_2}\right) + r \log \frac{r + \mu_2}{r + \mu_1} + \sum_{l=1}^{m} X_l \log \frac{p_{1l}}{p_{2l}}\right], \end{aligned}$$

where $X_+ = \sum_{l=1}^{m} X_l$. Finally, using the linearity of expectation and because $E[X] = \mu_1$ and $E[X_l] = \mu_1 p_{1l}$, we have:

$$D_{\mathrm{KL}}(f_1||f_2) = \mu_1 \log\left(\frac{r + \mu_2}{r + \mu_1} \frac{\mu_1}{\mu_2}\right) + r \log \frac{r + \mu_2}{r + \mu_1} + \sum_{l=1}^{m} \mu_1 p_{1l} \log \frac{p_{1l}}{p_{2l}}.$$

The symmetrized KL divergence used in Subsection 4.2.5, defined as the sum of the KL divergences from f_1 to f_2 and from f_2 to f_1, is given by:

$$D_{\mathrm{KL}}(f_1||f_2) + D_{\mathrm{KL}}(f_2||f_1) = (\mu_1 - \mu_2) \log\left(\frac{r + \mu_2}{r + \mu_1} \frac{\mu_1}{\mu_2}\right) + \sum_{l=1}^{m} (\mu_1 p_{1l} - \mu_2 p_{2l}) \log \frac{p_{1l}}{p_{2l}}.$$

Appendix C

Open-source Software

The software mentioned in the previous chapters is listed below. All programs are publicly available and released under an open-source license.

NucHunter Java program, described in Chapter 3, that infers nucleosome positions from one or multiple ChIP-seq BAM files. NucHunter can be downloaded from: `http://epigen.molgen.mpg.de/nuchunter/`.

bamsignals R package for computing count signals from BAM files. bamsignals is implemented mostly in C++ and uses the htseq library for reading BAM files efficiently. bamsignals is also a fundamental building block of EpiCSeg and Footifind. The package can be downloaded from Bioconductor [129]: `http://bioconductor.org/packages/release/bioc/html/bamsignals.html`.

kfoots R package for fitting the negative binomial and negative multinomial distributions, as well as mixture models and hidden Markov models based on them. kfoots is implemented mostly in C++ and uses multithreading for dealing with large datasets. The package can be downloaded from: `https://github.com/lamortenera/kfoots`.

EpiCSeg R package, described in Chapter 4, for computing chromatin state annotations from multiple BAM files. EpiCSeg is based on kfoots and bamsignals and offers several functions for analysis and visualization. EpiCSeg offers an R interface as well as a command-line interface. The package can be downloaded from: `https://github.com/lamortenera/epicseg`.

Appendix D

Summary

Proteins interacting with the genome, such as histones and transcription factors, play a major role in the regulation of gene expression. These interactions can be detected with ChIP-seq, which provides sequences of non-negative integers, called count signals, quantifying the presence of a given protein at each genomic locus. However, the computational analysis of count signals is challenging, as the biological patterns are complex and the datasets are large. In this thesis, we propose accurate and efficient algorithms for 3 different pattern detection problems in count signals.

First, we present an algorithm that infers the genomic locations of positioned nucleosomes from histone ChIP-seq experiments. This method can integrate measurements for different histone marks and uses a wavelet to detect the count pattern corresponding to positioned nucleosomes. When compared with previous approaches using biological and simulated data, our method shows a higher precision and reduced runtimes.

Next, we introduce an algorithm that annotates genomic regions according to the regulatory processes acting on them. The labels of this annotation, called chromatin states, are learned automatically from the measurements of multiple histone marks. Unlike previous approaches, our method characterizes chromatin states with a rigorous probabilistic model of the count signals. The resulting annotation is shown to be more strongly associated to DNA accessibility and transcription, as well as more robust and comprehensive compared to previous approaches.

Lastly, we present an algorithm for finding transcription factor binding sites from ChIP-exo data (a method similar to ChIP-seq). Our algorithm learns the genomic sequences that attract the transcription factor (the motif) and the count pattern observable at binding sites (the footprint) at once. We show that our method finds the correct motif and detects interpretable footprints in 4 different datasets. Moreover, our approach can distinguish different categories of binding sites in the same experiment.

Overall, the proposed algorithms represent an advancement in the automatic detection of biological patterns, as they are more accurate and in some cases considerably faster than existing approaches. Finally, they are based on a mathematical framework that is general and likely to be important for future research.

Appendix E

Zusammenfassung

Proteine, die mit dem Genom interagieren, spielen eine wichtige Rolle in der Regulation der Genexpression. Diese Interaktionen können mit Hilfe sogenannter ChIP-seq Experimente detektiert werden. Die resultierenden Messungen lassen sich durch Sequenzen von nicht-negativen ganzen Zahlen darstellen, die Zählsignale genannt werden und die die Proteinmenge in jedem Lokus quantifizieren. Die Analyse dieser Signale wird jedoch im Allgemeinen durch die Komplexität der biologischen Muster und der Größe der Datensätze erschwert. In der vorliegenden Arbeit werden Algorithmen für drei Mustererkennungsprobleme in Zählsignalen vorgeschlagen.

Als erstes wird ein Algorithmus präsentiert, der die Koordinaten gut positionierter Nukleosomen aus ChIP-seq Daten von Histonmodifikationen vorhersagt. Die vorgestellte Methode kann Messungen für verschiedene Histonmodifikationen integrieren und benutzt ein Wavelet um das Muster, das gut positionierten Nukleosomen entspricht, in dem Zählsignal zu erkennen. Ein Vergleich der vorgestellten Methode mit früheren Ansätzen auf biologischen sowie simulierten Daten zeigt, dass die neue Methode präziser und schneller ist.

Der zweite vorgestellte Algorithmus annotiert Genomregionen nach den auf sie wirkenden genregulatorischen Prozessen. Die Kategorien dieser Annotation, die Chromatinzustände genannt werden, werden automatisch aus den Messungen von mehreren Histonmodifikationen gelernt. Die vorgestellte Methode bestimmt Chromatinzustände mit Hilfe eines exakten Modells der Zählsignale. Die so gelernte Annotation ist besser mit Daten zur Genomzugänglichkeit und Transkription assoziiert, so wie robuster und umfassender im Vergleich zu früheren Ansätzen.

Als letztes wird ein Algorithmus beschrieben, der Bindungsstelle von Transktiptionsfaktoren aus einem ChIP-exo Experiment (eine ähnliche Methode wie ChIP-seq) vorhersagt. Der vorgestellte Algorithmus lernt gleichzeitig, welche Genomsequenzen die Transkriptionsfaktoren binden (das Motif) und welches Muster das Zählsignal an den Bindungsstellen zeigt (das Footprint). Auf vier unterschiedlichen Datensätzen wird gezeigt, dass die vorgestellte Methode immer das korrekte Motif und interpretierbare Footprints findet. Außerdem kann der vorgestellte Ansatz verschiedene Gruppen von Bindungsstellen in einem ChIP-exo Experiment erkennen.

Zusammenfassend, präsentiert die vorliegende Arbeit Methoden, die die bestehenden verbessern und die als Startpunkt fuer künftige Ansätze dienen können.

Appendix F

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich diese Arbeit selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel und Quellen verwendet habe. Ich erkläre weiterhin, dass ich die vorliegende Arbeit oder deren Inhalt nicht in einem früheren Promotionsverfahren eingereicht habe.

Berlin, den 30.10.2015 Alessandro Mammana