Fakultät für Informatik
Technische Universität München

Scalable longitudinal statistical models for genomics data

Georg Stricker

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender:
Prof. Dr. Florian Matthes

Prüfende der Dissertation:
1. Prof. Dr. Julien Gagneur
2. Prof. Dr. Thomas Huckle

Die Dissertation wurde am 25.06.2018 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 21.10.2018 angenommen.

Acknowledgments

First and foremost I'd like to thank Julien Gagneur, for supervising my project, being always strict with me on scientific rigour and his constant pushing for a better presentation and communication of scientific results. It's an under-appreciated and most useful skill to gain. I would also like to thank him personally for general guidance, a high level of flexibility and a very energetic and lively vibe that he created in his lab. It wasn't an easy decision to do a PhD, but one of the best I've ever made, which I would like to attribute to his scientific and personal character.

I am also most grateful to the entire lab, past and present for being a group of such amazing and highly mixed individuals. It was always a pleasure working, partying and arguing with you. Your patience with my humor and tendency to push your buttons and limits is highly appreciated. With most of you I grew past the state of colleagues to trues friends, which I'm sure will lead to countless encounters and future fiestas. Also I need a place to stay when I travel to you home country ;)

A special thanks goes out to any collaborator of the Gagneur Lab, but especially the Prokisch Lab and Soeding Lab, which fitted so well into the vibe with its wonderful people, that it is almost unbelievable, that we didn't naturally merge into a single lab at some point. Most of you are more then colleagues or collaborators.

Finally, I would like to thank my family, friends, Stephan Feldmeier and the entire team of KS-Gym and a very special teacher from my high school, who thought me the essence and value of education. You all shaped the person I am now. You are my home, my strength and my source of energy when I constantly bounce against the confined limits of my environment. Without you I'm incomplete.

Summary

Recent advances in sequencing techniques allow the measurement of various genomic information on whole genome level, such as protein occupancy, which is used to study DNA-protein interactions. Given a constantly increasing scientific focus on humans, innovations are not only required on the modeling side, but also must be capable of dealing with the size of the human genome, which accumulates to over 3 billion data points in a single experiment. So far common practice has been to focus on specific regions of interest (such as genes) or use metrics (such as binning) to reduce the complexity of the data, trading a gain in computational speed for a loss of information. However, new cutting edge techniques are already capable of measurements at single-nucleotide level. Thus advanced, sophisticated methods are required to make full use of this data within feasible computation time.

This thesis explores the development of a general statistical framework providing a robust basepair resolution fitting process based on longitudinal models. First, a statistical model is established for the analysis of data coming from experiments of chromatin immunoprecipitation followed by deep sequencing (ChIP-Seq). An approach widely used to study protein-DNA interactions. The quantities of interest are often the differential occupancies relative to controls, between genetic backgrounds, treatments, or combinations thereof. Although ChIP-Seq is a very generic methodology to study protein-DNA interactions, statistical analysis methods have been so far dedicated to specific applications. Hence, practitioners rely on different statistical frameworks for different tasks such as peak calling or differential binding. Additionally, current methods face a number of issues: The reliance on subjective sliding window techniques, a lack of additional data handling, such as replicates and control factors or limitation in statistical inference and proper statistical error handling.

The proposed method, termed GenoGAM (Genome-wide Generalized Additive Model), integrates the well-established and flexible generalized additive model framework into genomic applications using a data parallelism strategy. ChIP-Seq data is modelled jointly as a product of smooth functions along chromosomes. Smoothing parameters are objectively estimated from the data by cross-validation, eliminating ad-hoc binning and windowing needed by current approaches. Furthermore, it provides base-level and region-level significance testing for full factorial designs. Application to a ChIP-Seq dataset in yeast show increased sensitivity over existing differential occupancy methods while controlling for type I error rate. Extension to a peak caller and analysis of a set of DNA methylation data further illustrates the potential of GenoGAM as a generic statistical modeling tool for genome-wide assays.

Upon having established the statistical framework, the scalability and an efficient implementation of GenoGAM is developed in order to enable its use on all types of organ-

isms, especially large gigabase genomes like mouse and human. Approaches for solving linear systems are further explored, focusing in particular on exploiting the sparsity of the model. Combining the SuperLU direct solver, sparse Cholesky factorization and the sparse inverse subset algorithm yields a 2-3 orders of magnitude speedup. Furthermore the HDF5 library is employed to store data efficiently on the hard drive, reducing memory footprint while keeping I/O low. As a result whole-genome fits for human ChIP-Seq datasets (ca. 300 million parameters) can be obtained in less than 9 hours on a standard 60-core server. Moreover, the algorithmic improvements for fitting large GAMs could be of interest to the statistical community beyond the genomics field.

In conclusion this thesis establishes statistical models for long longitudinal data coming from genome-wide sequencing-based experiments. It is shown how generalized additive models can be effectively applied to model chromatin immunoprecipitation followed by sequencing data and how these models allow improved differential analysis and principled peak calling. Effective algorithms to fit these models to complete genomes combining approximated parallelization schemes and sparse matrix techniques are provided.

GenoGAM is implemented as an open source `R` package and available on GitHub and Bioconductor.

Publications

GenoGAM: Genome-wide generalized additive models for ChIP-Seq analysis

Ref. [1]
Georg Stricker*, Alexander Engelhardt*, Daniel Schulz Matthias Schmid, Achim Tresch and Julien Gagneur
* joint first author
(2017) Bioinformatics, DOI:10.1093/bioinformatics/btx150

Author contribution Conceived the project and supervised the work: JG AT. Developed the software and carried out the analysis: GS AE JG. Carried out the ChIP-Seq experiments for TFIIB on yeast: DS. Gave advice on statistics: MS. Wrote the manuscript: JG GS AE MS AT

GenoGAM 2.0: Scalable and efficient implementation of genome-wide generalized additive models for gigabase-scale genomes

Ref. [2]
Georg Stricker, Mathilde Galinier and Julien Gagneur (2018) BMC Bioinformatics, DOI:10.1186/s12859-018-2238-7

Author contribution Conceived the project and supervised the work: J.G. Developed the software and carried out the analysis: G.S., M.G. and J.G. Wrote the manuscript: G.S. and J.G.

Contents

1 Introduction

1.1 Biological background

1.1.1 Characterization of the genome

The genome is the complete genetic material of an organism that provides a genetic program instructing the cell how to behave. This genetic program consists of information which is encoded in the DNA: A structure of two complementary chains composed of sequences of the four nucleotides adenine (A), cytosine (C), guanine (G) and thymine (T) and bound together by hydrogen bonds into a double helix. In eukaryotic cells DNA is further compressed by wrapping around a protein core of eight histone molecules to form nucleosomes. Those nucleosomes are then further packaged into chromatin.

The knowledge of a complete genome provides the foundation of functional studies not only for the organism in question, but also other related organisms that share certain features. The first complete genome to be sequenced was *bacteriophage ΦX174* by Sanger et al. in 1977 [3]. A single-stranded DNA virus, which consisted of approximately 5,386 nucleotides. Further genomes of important model organisms followed in the years of the human genome project [4]. Most notably yeast (*Saccharomyces cerevisiae*, ∼12 megabases) in 1996 [5], the nematode worm (*Caenorhabditis elegans*, ∼100 megabases) in 1998 [6], the fruit fly (*Drosophila melanogaster*, ∼120 megabases) in 2000 [7] and mouse (*Mus musculus*, ∼2.8 gigabases) in 2002 [8]. Finally, the completion of human genome sequencing in 2003 marked a major milestone in the history of modern biology [4]. With the complete sequence at hand the focus turned to the annotation of genomes for functional content, including protein-coding and non–protein-coding genes, transcriptional regulatory elements, and sequences that mediate chromosome structure and dynamics [9]. In particular it opened the door for characterization of the physical genome, which is relevant to many different fundamental processes such as transcription, DNA replication and repair, recombination and chromosome segregation.

1.1.1.1 Transcription factor binding

The ability of the cell to carry out those processes accurately relies on a complex, multilevel regulatory system. At the lowest level, the transcription level, those processes are controlled by regulatory elements that recruit transcription factors with specific DNA recognition properties [10] (see Figure 1.1a). But even if the transcription factors are known, their target sites or regulated target gene or element might be not. Genome-wide studies can help reveal the connection between transcription factor binding site (TFBS),

transcription factor (TF) and target element (e.g a gene) [11, 12]. Generally, those connections may vary in different ways: by distance, by specificity and by role in the binding process. Binding sites can be located locally at promoter regions just upstream of the target gene [13, 14, 15, 16] or at distant enhancer regions up to 1Mbp (1,000,000 bp) upstream or downstream of the transcription start site (TSS) [17, 18, 19, 20] (see Figure 1.1a and b). The specificity across TFBS can vary, with certain loci bound more frequently than others despite the same consensus sequence [16, 21, 22, 23]. A TF might not interact directly with DNA, but instead be involved in the recruitment process of other proteins [24, 25] (see Figure 1.1b-d). Furthermore determining the binding sites of these regulatory proteins in the genome is important for reconstruction of transcriptional regulatory networks, which can identify global chromosomal features or periodicity in interactions with the genome [26, 27, 28, 29, 30].

1.1.1.2 Histone modification

The behaviour of transcription factor binding in different tissues or pathways shows the influence of additional factors beyond DNA sequence alone, such as chromatin structure and its core component, the nucleosome. The link between transcription factors and chromatin has been known for a long time [31]. However, study of this interaction has been proven difficult until the emerging knowledge of complete genome sequences and techniques to study protein-DNA interactions *in vivo*. That is, studies in which the effects of various biological entities are tested on whole, living organisms or cells. As opposed to a tissue extract or dead organism, which is called *in vitro*. In particular the tails of the eight core histones are subject to enzyme-catalyzed manipulations and modifications through acetylation, phosphorylation, methylation and ubiquitination. An attachment of an acetyl, phosphoryl, methyl or ubiquitin group, respectively, that functions as a marker for other molecular elements. Because nucleosomes are directly involved in the packaging structure of DNA, various models have been proposed for functional mechanisms between histone modifications and gene regulation [32, 33, 34, 35, 36, 37]. Moreover interplay between a combination of histone modifications add to the complexity of the epigenetic code [38]. If DNA is tightly wrapped around the histones, it is generally hard for other proteins to access and interact with it. Although this makes a direct study of those DNA regions difficult, the very presence of histones at those particular regions contains valuable information. They can be seen as marks for the cell, associated with certain processes such as transcription of active genes [39, 40, 25, 41] (see Figure 1.1).

1.1.1.3 DNA methylation

Another important regulator of gene expression is DNA methylation. Methylation occurs at cytosine and adenine bases, though cytosine methylation is more widespread and of greater importance for mammals [43, 44]. Although adenine has been observed as well [45]. Most CpG dinucleotides (cytosine nucleotide sequentially followed by a guanine nucleotide) are subject to methylation except for *CpG islands* that are mostly associated

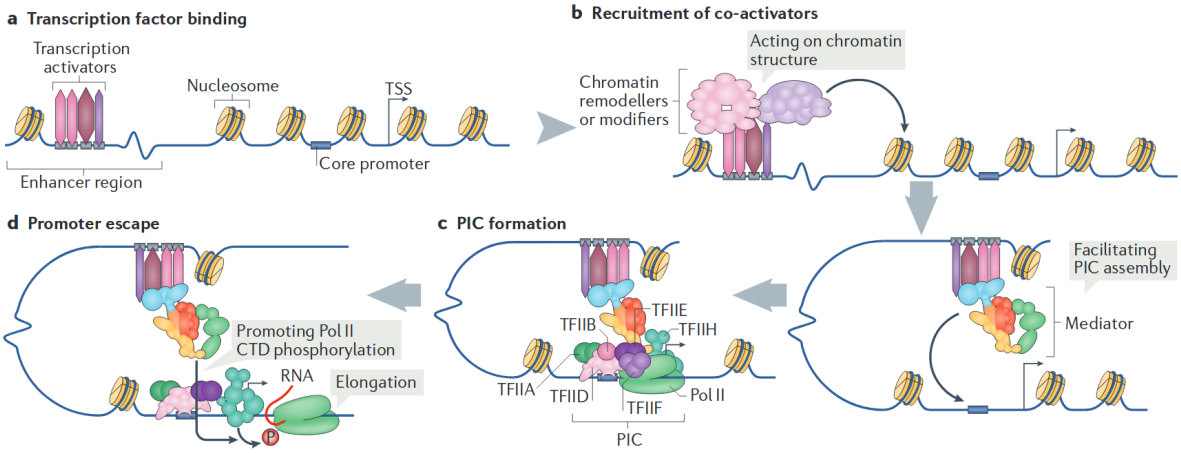

Figure 1.1: Transcription activation by RNA polymerase II. Taken from Ref. [42], Figure 1. A simplified model for the main steps of transcription initiation by Pol II in a chromatin context. Of note, the size and proportions of the depicted components do not reflect their actual dimensions. **(a)** Transcription activation starts with the binding of transcription factors (in this case, activators) on enhancer regions. These enhancer regions are located at different distances from the core promoters. The TSS is indicated by an arrow to the right. **(b)** Activators then recruit co-activator complexes that act as chromatin modifiers or remodellers to alter chromatin structure and to make it more accessible for other factors. Other co-activators are then recruited that act directly on the assembly of basal transcriptional machinery, the so-called preinitiation complex (PIC). Many co-activators act in cooperation, and some have functions both as chromatin regulators and as co-activators, contributing to PIC formation. The functions of chromatin regulators are not depicted in detail. In general, transcriptional co-regulators transmit the regulatory signals from the specific transcription factors to the PIC. Mediator of RNA polymerase II transcription (Mediator) is one of the key co-activator complexes. **(c)** The PIC is assembled at the core promoter. It includes Pol II and a number of general transcription factors. Multiple steps and pathways could be involved in PIC assembly in vivo, and Mediator acts to facilitate recruitment and/or stability of different PIC components. **(d)** Phosphorylation (P) of Pol II is necessary for Pol II to escape from the promoter and for the transition from the initiation step to the elongation step (creation of an RNA copy from DNA).

with functional gene promoters [46]. CpG islands are usually C+G rich regions (content above 50%) with a length greater than 200bp and a ratio of observed to expected CpG greater than 0.6. Here, the *ratio of observed to expected* refers to the presence of CpG pairs (observed) compared to the independent presence of C and G alone (expected). A ratio of 1 would thus mean that within a genomic region the C and G content is solely due to the presence of CpG pairs. Hypermethylation of CpG islands frequently contributes to the development of cancer by silencing tumor suppressor genes [44]. Additionally research has shown that extensive links and crosstalk between histone modifications and DNA methylation exists [47].

1.1.2 A short history of ChIP-Seq

1.1.2.1 Chromatin immunoprecipitation (ChIP)

Since transcription factors are a key vehicle to regulate gene expression, many approaches have been developed to identify their target sites *in vivo*. Among the first were direct footprinting studies of targets in promoter regions upstream of the gene, which allowed only for a handful of promoters to be examined at once [48]. Alternative methods examining transcription profiles over all known genes in the absence of a particular transcription factor or expression analysis of mutants using microarrays enabled exploration on whole-genome level [49, 50, 51]. However, those methods suffered from an indirect approach, resulting in uncertain conclusions about the nature of change (or lack of it) in transcription levels.

One of the first methods to successfully identify bound transcription factors directly in a whole-genome analysis was ChIP-chip ([21, 12], see Figure 1.2), originally developed by Horak and Snyder in 2002 for analysis in yeast [52]. It makes use of the idea of ChIP, where proteins are crosslinked with DNA, fragmented by sonication and isolated by immunoprecipitation (IP) to obtain chunks of DNA sequences, where the bound transcription factor has been identified by an antibody. This idea goes back to Solomon and Varshavsky [53], who used formaldehyde (HCHO) to stabilize a variety of crosslinks with virtually no reactivity towards free double-stranded DNA [54], opposed to a treatment with other agents like UV-light or dimethyl sulfate [53]. In an later experiment Solomon additionally used IP to examine *in vivo* the interaction between histone H4 and heat shock protein 70 (hsp70) genes. Contrary to previous *in vitro* studies he could show that although transcription perturbs nucleosome structure, at least histone H4 remains bound to actively transcribed DNA sequences [55]. Further development, including amplification by Polymerase chain reaction (PCR), was done by Dedon [56, 57], Orlando [58, 59] and Strutt [60], but had its major breakthrough through the works of Hecht and Strahl-Bolsinger [61, 62].

An alternative method to identify DNA loci which interact with proteins is DNA adenine methyltransferase identification (DamID), developed by van Steensel and Henikoff [63]. It is a methylation-based tagging technique in which DNA adenine methyltransferase (Dam) is fused to a protein of interest resulting in local DNA methylation at native binding sites of the protein. Methylation-specific restriction enzymes or antibod-

ies are then used to map the regions. Therefore DamID bypasses the dependence on antibodies and the need to chemically crosslink DNA–protein complexes. Although the former is true, ChIP techniques can easily be generalized to proteins lacking antibodies by tagging them with epitopes for which high quality antibodies are available. By contrast, DamID is of limited value for discriminating binding of a given factor depending on its post-transcriptional modifications status [64].

The success of ChIP and its combination with microarrays (ChIP-chip) led to a number of developments combining it with various other techniques to expand genome-wide studies to more complex interactions and genomes (GMAT [65], ChAP [66], SACO [67]). In particular strategies of ChIP in combination with sequencing have demonstrated a way to overcome the limited scale of the original ChIP-chip protocol (STAGE [68], ChIP-PET [69], MS-PET [70]).

1.1.2.2 Chromatin immunoprecipitation followed by massively parallel DNA sequencing (ChIP-Seq)

Rapid technological developments in next-generation sequencing (NGS) enabled increasingly large experiments. New sequencing platforms of Solexa/Illumina and 454 allowed to sequence greater numbers of DNA fragments faster and cheaper than before. In 2007 several labs were developing a protocol that could be used in concert with ChIP to produce high-quality protein-DNA interactome measurements, ultimately terming it chromatin immunoprecipitation with massively parallel DNA sequencing (ChIP-Seq) ([72, 73, 74, 75, 76], see Figure 1.3). From the high number of DNA fragments produced by the new sequencing platforms through the ChIP protocol, short reads could be determined and then mapped onto the reference genome, calculating the frequency of the protein of interest in the sample. In contrast to ChIP-chip, ChIP-Seq yields higher resolution, fewer artefacts and a greater coverage. This in turn improves the characterization of DNA-binding proteins and makes identification of sequence motifs possible. In particular profiling of nucleosome-level features profits from increased precision and allows a better systematic cataloguing of patterns of histone modifications and nucleosome positioning.

As a successor to ChIP-chip, ChIP-Seq inherits some of its disadvantages. The crucial dependence on antibody quality remains by nature of the experimental design. As well as the bias towards GC-rich content in fragment selection, both in library preparation and in amplification before and during sequencing [77]. Another bias occurs during fragmentation of the genome: Open chromatin regions tend to be fragmented more easily than closed regions, leading to a non-uniform distribution of reads. Two strategies are essentially used to tackle those biases: The most widely used is input DNA, where a portion of the DNA sample skips the IP step jumping directly to library preparation. Therefore input DNA is cross-linked and fragmented under the same conditions as the IP DNA. Additionally mock IP DNA (DNA obtained from IP without antibodies) can be used to control for over representation of open chromatin regions [78, 79]. Of those controls, input DNA seems to be the most efficient, while the mock IP is found to contribute little to the overall result when the data is properly normalized [80].

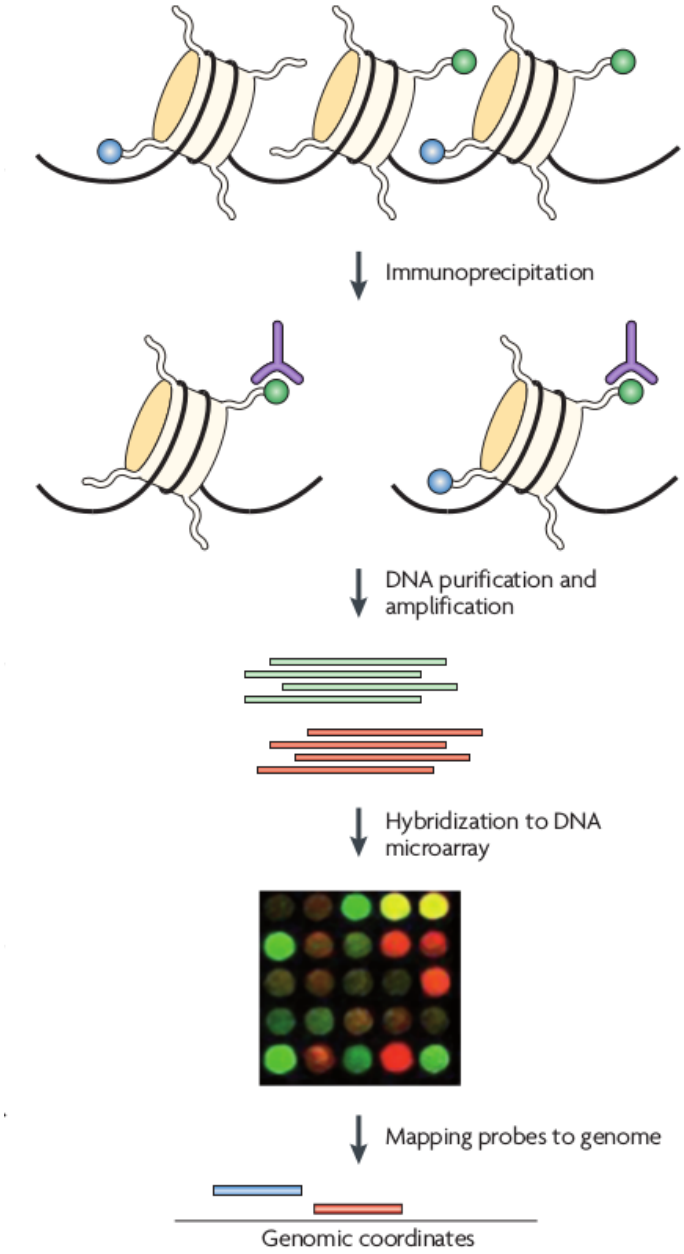

Figure 1.2: Chromatin immunoprecipitation combined with DNA microarrays (ChIP-chip). Taken from Ref. [71], Figure 2. This figure shows how ChIP-chip is used to study histone modifications. Modified chromatin is first purified by immunoprecipitating crosslinked chromatin using an antibody that is specific to a particular histone modification (shown in green). DNA is then amplified to obtain sufficient DNA. The colour-labelled ChIP DNA, together with the control DNA prepared from input chromatin and labelled with a different colour, is hybridized to a DNA microarray. The microarray probes can then be mapped to the genome to yield genomic coordinates.

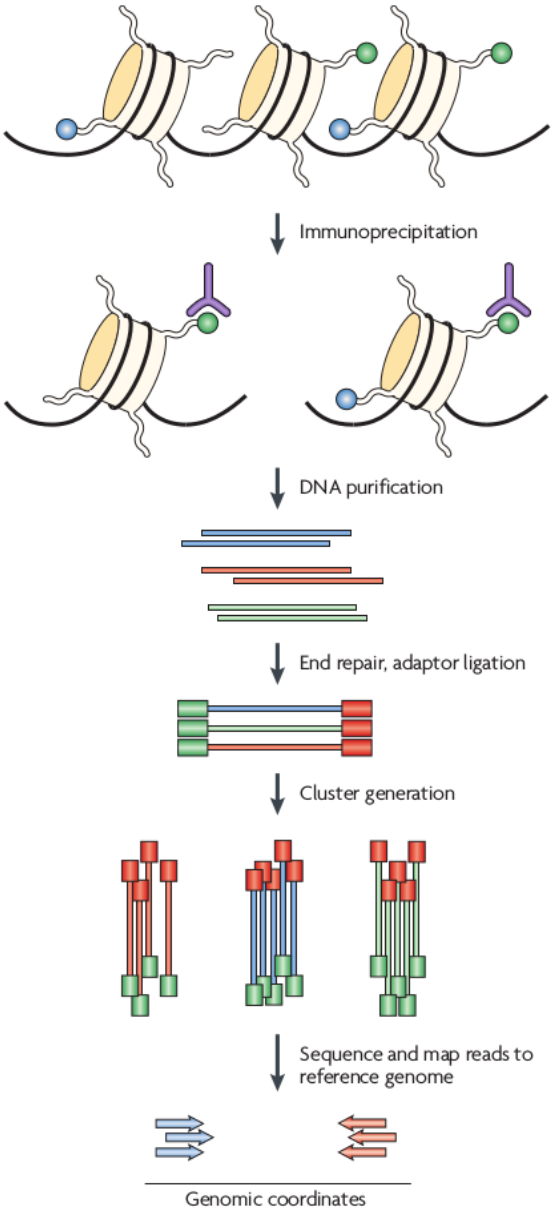

Figure 1.3: Chromatin immunoprecipitation combined with high-throughput sequencing techniques (ChIP-Seq). Taken from Ref. [71], Figure 4. This figure shows how ChIP-Seq is used to study histone modifications. The first step is the purification of modified chromatin by immunoprecipitation using an antibody that is specific to a particular histone modification (shown in green). The ChIP DNA ends are repaired and ligated to a pair of adaptors, followed by limited PCR amplification. The DNA molecules are bound to the surface of a flow cell that contains covalently bound oligonucleotides that recognize the adaptor sequences. Clusters of individual DNA molecules are generated by solid-phase PCR and sequencing by synthesis is performed. The resulting sequence reads are mapped to a reference genome to obtain genomic coordinates that correspond to the immunoprecipitated fragments.

In order to perform meaningful downstream analysis of ChIP-Seq data, a sufficient number of reads has to be sequenced. This is especially true for input DNA that does distribute more uniformly along the genome, contrary to IP DNA, which clusters around the binding sites. Thus, in order to obtain a useful control sample, sequencing depth of at least the same size as the IP is required [79], greatly increasing the overall number of reads needed. Fortunately, in recent years, as further improvements in NGS were made and adaptation of ChIP-Seq grew, obtaining a high number of reads has become less of an issue, even for larger organisms like human.

One of the advantages of ChIP-Seq over ChIP-chip is the improved resolution (see Figure 1.4b and c, Figure 1.7a). However, size heterogeneity of randomly sheared ChIP DNA limits mapping resolution. In recent years further research have lead to advanced techniques to address this issue in the development of ChIP-exo [81] (see Figure 1.4) and ChIP-nexus [82]. Both methods enhance ChIP-Seq by treating ChIP DNA with exonuclease during immunoprecipitation, cutting non-specific DNA left and right of the bound protein (Figure 1.4a). This step leads to an overall higher technical complexity and lower amount of recovered DNA, however resulting in a less robust experimental setting [82]. ChIP-nexus improves the efficiency of library preparation and adds a unique, randomized barcode to the adaptor to monitor overamplification during PCR [82]. Despite this advances adaptation of both protocols has been slow. Some applications do not require base-pair resolution or can achieve sufficient results through computational methods. Hence, the cost of adaptation and an increase in technical complexity ends up dominating the advantages. Therefore, over 10 years after the introduction of ChIP-Seq, it remains the most widely used protocol for quantification of protein-DNA interaction.

1.2 Computational background

1.2.1 ChIP-Seq workflow

Computational methods have always accompanied development in molecular quantification methods, becoming increasingly important with growing amounts of data. Thus, huge efforts have been made to improve tools for analysis of data and create computational pipelines. It can be roughly divided into seven steps, for which numerous specialized software have been developed: Pre-analysis steps like mapping reads and computation of quality metrics for sequences and read counts. Preprocessing steps, such as determination of fragment size, transformation of count data by binning, smoothing and filtering. Normalization to account for different sequencing depth. Identification of enriched regions or peak calling. Computation and multiple testing correction of significance values. Differential binding analysis for multiple treatments and downstream evaluation of results, such as peak annotation, motif analysis and motif discovery [83] (Figure 1.5). Which of those steps will be performed greatly depends on the question of interest. For example, differential binding analysis of genes does not require identification of enriched regions, as they are known prior. Whereas identification of differentially bound sites does require determination of such sites first. Throughout the development

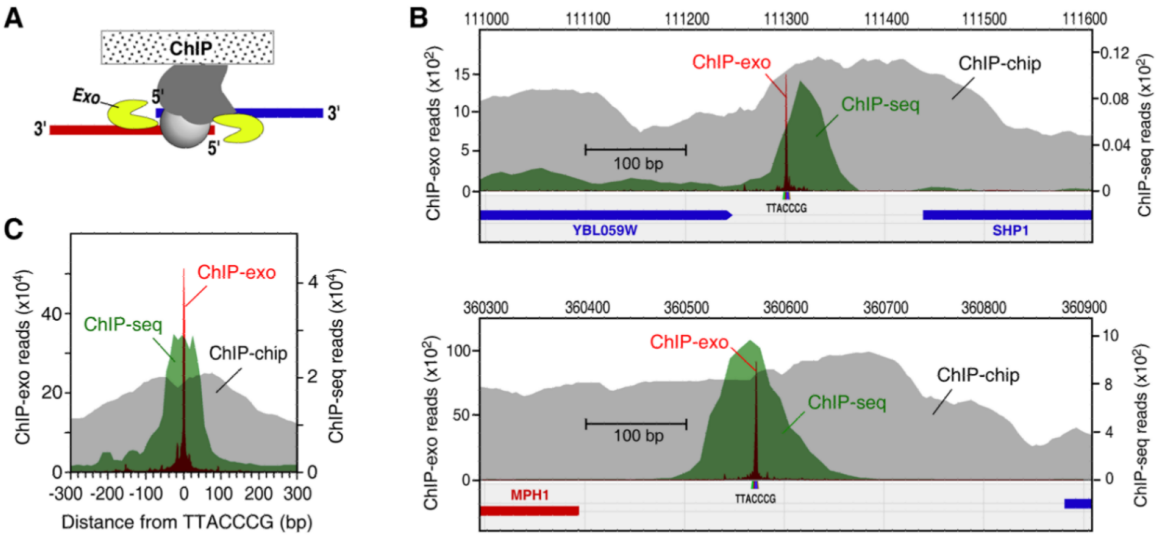

Figure 1.4: Chromatin immunoprecipitation combined with high-throughput sequencing and exonuclease treatment (ChIP-exo). Taken from Ref. [81], Figure 1. **(A)** Illustration of the ChIP-exo method. ChIP DNA is treated with a 5' to 3' exonuclease (yellow) while still present within the immunoprecipitate. **(B)** Comparison of ChIP-exo (red) to ChIP-chip (gray) and ChIP-Seq (green) for Reb1 at specific loci. The plots show the distribution of raw signals. **(C)** Aggregated raw Reb1 signal distribution around all 791 instances of the motif `TTACCCG` in the yeast genome.

stages some steps, like normalization became indispensable. Thus, nowadays most software implement an entire pipeline for the question of interest rather than a single step within the pipeline.

1.2.1.1 Mapping and sequence alignment

Mapping reads is the procedure to find the most likely position of a certain read in the reference genome. It is a complex process that has to take into account the numerous variations of individual DNA deviating from the reference genome. Thus, each software provides a mechanism to compute a mapping score that contains information about how well the read fits an identified position. Additionally, organisms with a high fraction of repetitive elements in their genome pose a particularly challenging task to mapping software. The human genome, for example is estimated to contain around 66% of repetitive elements [84]. The landscape of DNA mappers is relatively narrow and has been widely dominated by Bowtie [85] and BWA [86], but was recently joined by the RNA-Seq aligner STAR [87] that can be easily used for DNA mapping as well. Once the reads are mapped, their quality is assessed and low-quality reads are discarded or marked, such that they can be accounted for. Usually candidates are checked for mapping quality or library complexity due to antibody quality or PCR over-amplification.

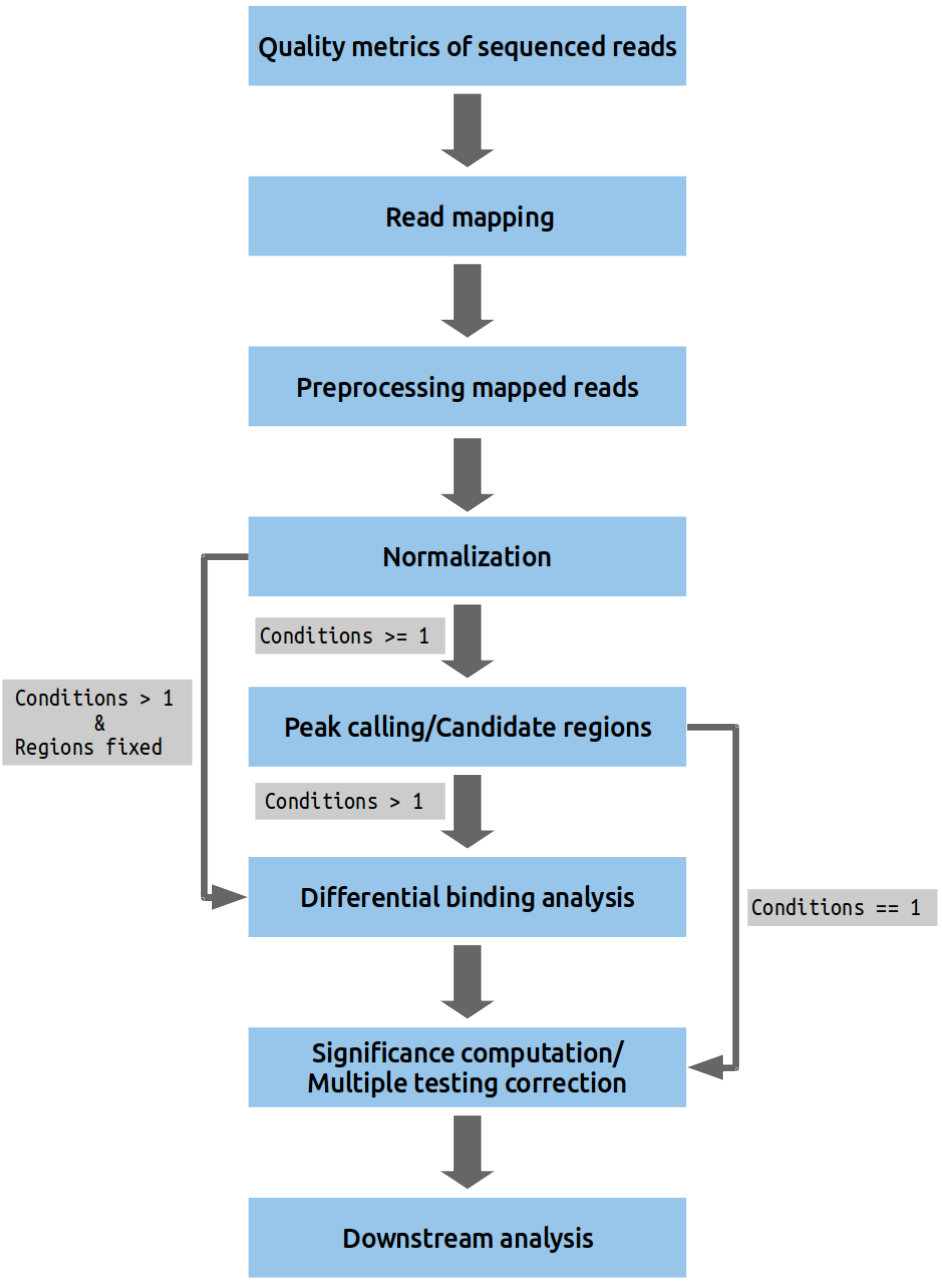

Figure 1.5: Workflow for computational analysis of ChIP-Seq. Similar to Figure 1 in Ref. [83]. A typical software for ChIP-Seq starts at the third box from the top (Preprocessing). It is usually meant to do one particular analysis: Peak calling in the IP sample with or without control (right grey arrow). Differential binding analysis in IP samples under multiple conditions on prior given regions (left grey arrow). Differential binding analysis in IP samples under multiple conditions without prior given regions (middle grey arrow). In the latter case, the peak caller functions as the identifier of enriched regions, which are then given to the differential binding algorithm. Methods which employ this complete pipeline themselves, usually don't call peaks like a (narrow) peak caller would, but rather identify candidate regions more similar to a broad peak calling approach.

1.2.1.2 Preprocessing

Once raw short reads (*tags*) are mapped and filtered for quality they have to be preprocessed and quantified into proper format. In ChIP-Seq DNA is usually cut randomly in fragments of pre-selected size, of which only one or both short ends are sequenced. In case that both ends are sequenced (*paired end sequencing*), the fragment can be exactly identified. In case that only one end is sequenced (*single end sequencing*) strategies have to be applied to determine the entire fragment. The most widely used strategy is strand cross-correlation (Figure 1.6). The assumption is that tags on both strands accumulate on average around half a fragment length upstream of the actual binding site (equivalent to the fragment being centered on the binding site) and thus form related peaks downstream of the opposite strand (Figure 1.6 A and B). Therefore, the highest correlation for a relative shift between two such peaks on opposite strands indicates the offset to the binding site of the protein of interest (Figure 1.6C). Hence, the distance between those peaks (equals two times the shift) constitutes the fragment size and the center possibly a TFBS.

Often a sliding window approach is employed to transform overlapping positions of short reads into counts. But the methodology widely varies how this is performed in the pipeline: SISSRs [88] uses tags to directly identify TFBS on the fly by subtracting minus strand reads from plus strand reads within a window and mark positions of sign change. Some methods shift reads by the determined offset before counting them [89, 90, 91]. While other extend reads to full fragments first [92, 93, 94]. A few methods compute fragment length but nevertheless conduct most of the computation on the tags directly [95, 96].

In principle, analysis could be performed at every single base of the genome. However, the per-base coverage is in practice too low and too noisy for such an approach to have enough statistical power. Testing for enriched sites has been therefore done by integration of data over regions into windows (overlapping) or bins (consecutive). For convenience both terms will be referred to as *windows* hereafter and specified if they overlap or not only if required. Each fragment or read can theoretically overlap multiple windows. Depending on the method it is either assigned to at most one [97] or multiple windows [98, 99]. The total number of overlaps constitutes the window coverage. Alternatively coverage is computed per base and sequentially summarized into windows [100]. Those approaches are often coupled directly with a filter that discards low count windows if they fall below a certain threshold. The threshold can be based on estimation of a constant [101], a dynamic background signal [102], a probability obtained by simulating random read distributions [100], or presence of artifacts, such as single tag peaks [103] (an accumulation of tags on only one strand, without corresponding tags on the other strand).

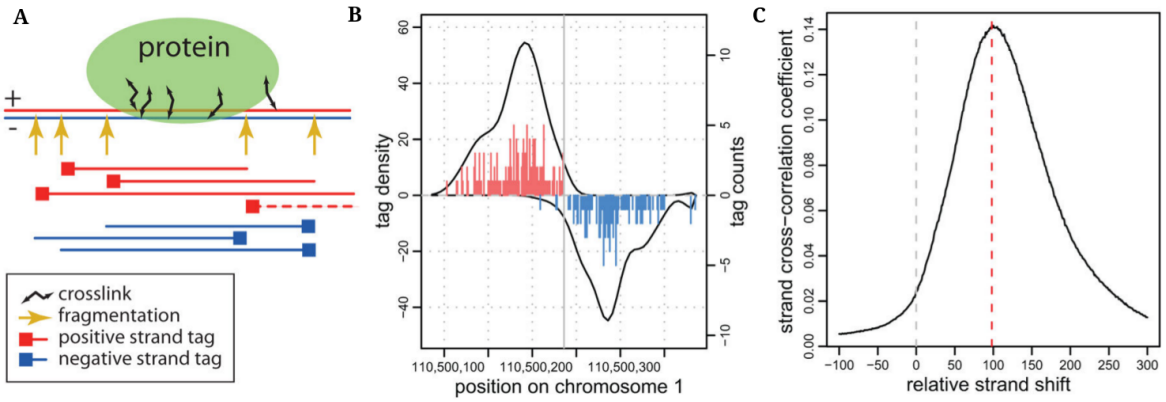

Figure 1.6: Concept of strand cross-correlation. Taken from Ref. [95], Figure 1. Panel A was removed and other panel letters adjusted accordingly. **(A)** A schematic illustration of ChIP-Seq measurements. The protein of interest (green) is bound to a DNA fragment. The 5' ends (squares) of the selected fragments are sequenced, typically forming groups of positive and negative strand tags on the two sides of the protected region. The dashed red line illustrates a fragment generated from a long cross-link that may account for the tag patterns observed in CTCF and STAT1 datasets. **(B)** Tag distribution around a stable NRSF binding position. Vertical lines show the number of tags (right axis) whose 5' position maps to a given location on positive (red) or negative (blue) strands. Positive and negative values on the y-axis are used to illustrate tags mapping to positive and negative strands respectively. The solid curves show tag density for each strand (left axis). **(C)** Strand cross-correlation for the NRSF data. The y-axis shows Pearson linear correlation coefficient between genome-wide profiles of tag density of positive and negative strands, shifted relative to each-other by a distance specified on the x-axis. The peak position (red vertical line) indicates a typical distance separating positive- and negative-strand peaks associated with the stable binding positions.

1.2.1.3 Normalization

Although in the very beginning following the ChIP-Seq development methods analyzed the IP sample only [76, 74, 96], having an additional control sample became standard procedure very quickly (see Figure 1.7a, bottom). Since then, more complex experimental designs evolved involving samples with different treatments and biological replicates resulting in the need to normalize data to make it comparable. An intuitive and widely used method to control for sequencing depth is linear normalization, where samples are multiplied by a scaling factor [73]. However, the computation of the factor varies from method to method. The most simple way to derive a normalization factor is by using the library size, i.e. the total number of fragments in a sample [95, 104, 103]. A more sophisticated way in order to account for local fluctuations and outlier regions is to use a representative sample of the fragments to avoid any bias from overrepresented regions. For instance, PeakSeq excludes enriched candidate windows to estimate a slope by linear regression on low count regions, which it regards as background [100]. Thus, excluding bound regions that might bias the baseline normalization factor. In contrast, diffReps calculates the scaling factor on enriched windows [101]. The reasoning is that background coverage might be too low to estimate a proper factor. Hence, using only abundant regions might yield a more reasonable scaling factor. A compromise is offered by the trimmed median of M-values approach originally developed for RNA-Seq [105]. Here, windows with highest and lowest log fold change are discarded to estimate the normalization factor on regions with moderate log fold change [98].

Furthermore, variations exists that try to approximate the scaling factor with a higher precision: Iterative methods estimate the background threshold and the scaling factor jointly in a greedy fashion [106, 107], compute regression lines on a mean-difference plot [108] or use expression levels of housekeeping genes [109]. Alternatively, non-linear normalization methods account for positional biological variations by estimating scaling factors locally. Motivated by the mixed binding pattern expressed by RNA polymerase II (Pol II) normalization methods based on Local Regression (LOESS) were developed [110, 111]. Other methods employ sigmoid functions [112] inside each window or scale the read coverage distribution of the IP sample to have the same mean and variance as the control sample [113] and adjust each window accordingly.

1.2.1.4 Peak calling

After the data has been smoothed and normalized to make different samples comparable the IP signal is controlled for background noise from the control sample or other sources of bias such as mappability or GC content [114, 97]. Two basic approaches exist to control for an input sample: subtract or divide the normalized IP signal by the normalized control signal and perform peak calling [109, 114, 115]. Or perform peak calling first and then use probabilistic measures to compute the significance of the peak given the control signal (see next subsection). In case a control sample does not exist, background signal is estimated directly from the IP sample by simulation. The assumption is that low count regions with no bound protein contain signal due to background noise. Therefore,

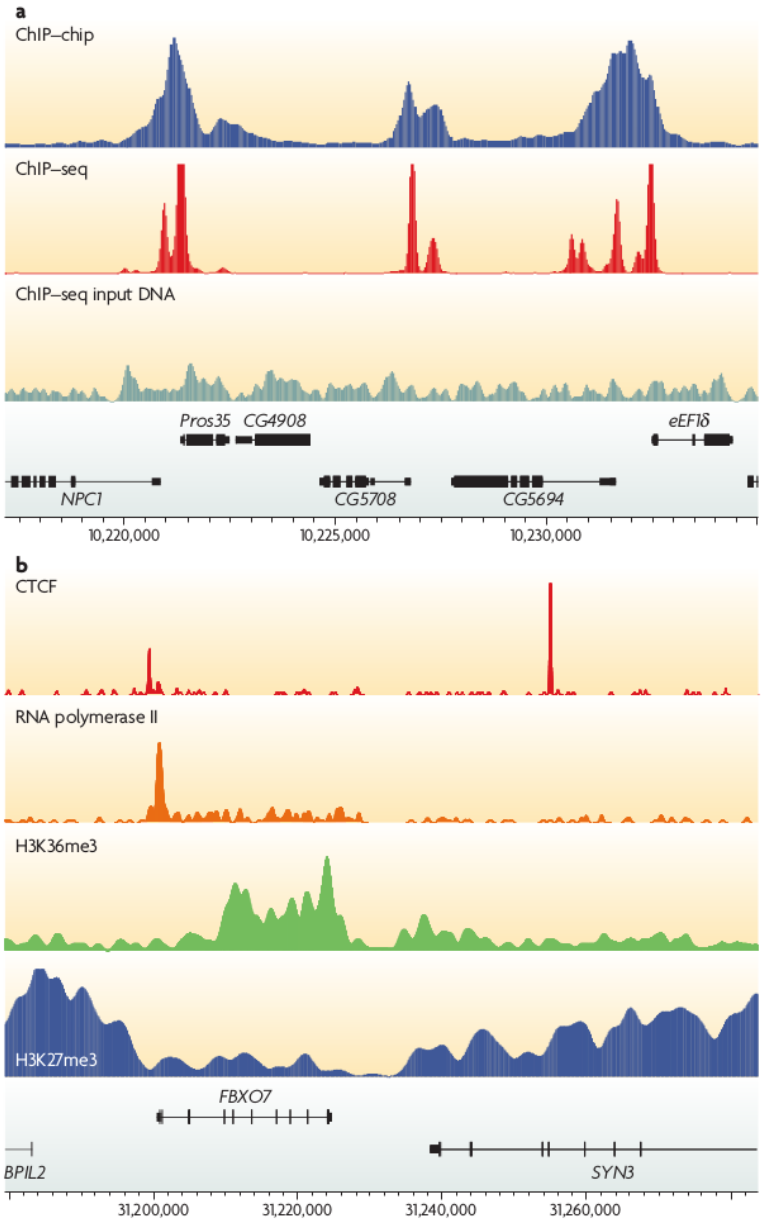

Figure 1.7: ChIP profiles. Taken from Ref. [78], Figure 2. **(a)** Shown is a section of the binding profiles of the chromodomain protein Chromator, as measured by ChIP-chip (blue) and ChIP-Seq (red) in the Drosophila melanogaster S2 cell line. The tag density profile obtained by ChIP-Seq reveals specific positions of Chromator binding with higher spatial resolution and sensitivity. The ChIP-Seq input DNA (control experiment) tag density is shown in grey for comparison. **(b)** Examples of different types of ChIP-Seq tag density profiles in human Tcells. Profiles for different types of proteins and histone marks can have different types of features, such as: sharp binding sites, as shown for the insulator binding protein CTCF (red); a mixture of shapes, as shown for RNA polymerase II (orange), which has a sharp peak followed by a broad region of enrichment; medium size broad peaks, as shown for histone H3 trimethylated at lysine 36 (H3K36me3; green), which is associated with transcription elongation over the gene; or large domains, as shown for histone H3 trimethylated at lysine 27 (H3K27me3; blue), which is a repressive mark that is indicative of Polycomb-mediated silencing.

Monte Carlo simulations are run to estimate the count distribution given that reads could be mapped randomly to the genome [96, 100, 113]. However, often the resulting background signal is not expressed in terms of a baseline threshold, but in terms of a distribution, that is sequentially used to compute the False Discovery Rate (FDR) of a given peak.

Peak calling is usually a two-step process. First identifying enriched windows and subsequently merging them into intervals to form peaks (Figure 1.8). It is important to note, that *peak* refers to a region of possibly several hundred basepairs which contains at least one *summit* (a single position with highest count in the vicinity). Therefore, a third step is sometimes employed to identify all subpeaks within peak regions or refine the peak to a narrow vicinity of the summit (Figure 1.8 last step). In particular, narrow peak callers make use of this step to identify sharp peaks (Figure 1.7b, top, red) in contrast to broad peaks (Figure 1.7b, middle, green) and entire domains (Figure 1.7b, bottom, blue).

Because the first step is concerned with filtering candidate regions from noisy background regions, probabilistic models are employed that can provide a measure of significance. Methods that apply sliding windows techniques often base their threshold on the distribution of the background signal, calling all regions lower than a certain FDR or p-value [103, 100, 104]. Alternative methods make use of Hidden Markov Models (HMM) [115, 116] or mixture models [97] to cluster windows into background and enriched windows or combination thereof [117]. In differential binding analysis it is common to divide the category of enriched windows further into the two categories enrichment in first sample and enrichment in the second sample [109] (Figure 1.9). Close or overlapping candidate windows are merged to form candidate peaks (narrow peak calling) or larger domains (broad peak calling), which can be scanned for further subpeaks, for example using kernel methods [118, 119, 120].

1.2.1.5 Differential binding analysis

In contrast to peak calling, differential binding analysis is concerned with identification of enriched regions between two IP samples of which one received a different treatment. For instance studying the cross-talk mechanism between histones by truncation of a core enzyme [121]. Examining the epigenomic effects of cocaine on mouse nucleus accumbens [122] or conducting an analysis of regulatory genomic features in lymphomas comparing follicular lymphoma cells (FLs) with populations of B cells from healthy donors [123]. As well as further numerous experiment designs enabled by the most recent rise of CRISPR/Cas9, following its relatively precise and cheap gene editing ability.

Since the computational task remains the same peak callers can be used to identify regions by replacing the control sample with the second IP sample. However this prevents the possibility of normalization by control. Furthermore, peak callers are designed for one-way enrichment, that is one sample over the other. Whereas in differential binding enrichment usually occurs in both directions. Additionally, regions of interest might be known prior (e.g. genes) and thus there is no need for identification of regions, but rather correct handling of type I error rates. Therefore special methods were developed

adapting strategies from broad peak callers (if regions are not fixed) [99, 109] or methods originally developed for differential expression analysis in RNA-Seq data (if regions are fixed) [124, 98, 125]. Alternatively to the former methods, the latter are used in concert with peak callers where peaks are supplied as fixed regions to get tested for differential occupancy.

1.2.1.6 Significance computation

Finally, all identified regions are evaluated for significance and ranked accordingly. A common way is to construct p-values from the underlying distributions used in the peak calling step and correct them for multiple testing to obtain FDR values. In case no control sample is available peak calling methods compute FDR values by constructing a null distribution by Monte Carlo simulation and testing against it [103, 100]. Otherwise the control sample is swapped with the IP sample to call *negative peaks* which serves as the set of true negatives in the computation of FDR [104, 106, 93]. Alternatively, positionwise permutation between samples can be performed to obtain a null distribution [113]. Differential binding methods that originated from RNA-Seq usually take a count matrix as input where samples are the rows, regions are the columns and the cells are the sums of read counts over given regions. Therefore, above techniques for raw read data can't be used. Instead, they use a regression approach over all regions, sharing information, like distributional parameters, between genes based on the assumption that data from different genes follow similar patterns of variability [124].

1.2.1.7 Further downstream analysis

In the early years of ChIP-Seq analysis many methodological advances were concerned with peak calling and differential occupancy. However, the approximate knowledge of binding positions by itself is of little value, as it does not offer insights into the functionality of the underlying sequence. Therefore, called regions are used as input for further analysis like motif discovery or annotation. Specific tools look for close associations with known sequences or positions from literature, like promoter regions or repeatedly occurring sequence patterns in the peak region that might indicate a new motif [126, 127, 128]. Another important task is to learn the sequence specificities of binding proteins, an information essential to build models for regulatory processes. With the recent rise of deep learning techniques, there has been growing interest in its application to genomics data [129, 130, 131], where the initial set of sequence features are extracted by peak callers. Therefore, despite the focus shifting to other challenges, ChIP-Seq remains an established technique widely used in concert with other methods as the first step in the analysis pipeline to study gene regulation.

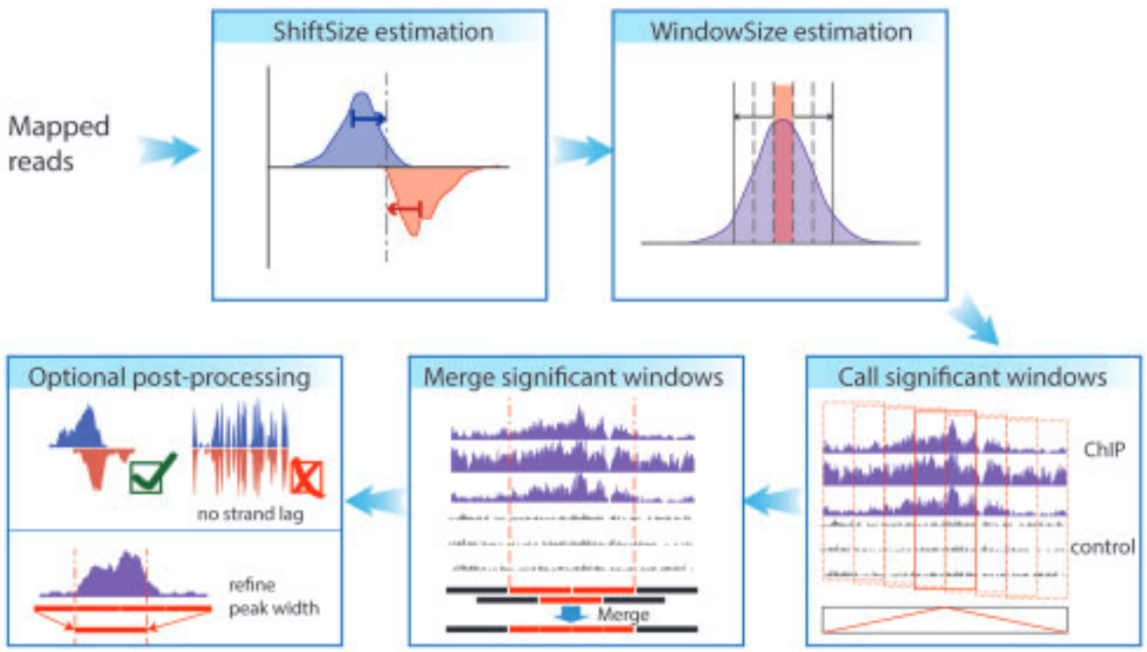

Figure 1.8: Example workflow for identification of enriched regions and differential binding. Taken from PePr, Ref. [99], Figure 1. A tag shift is estimated and all reads are shifted. Raw read data is then used to identify regions of abundance. The average peaks width of the top abundant regions are used to estimated a window size. The data is the split into windows overlapping by 50%. For each window the sum of counts is modeled with a local negative binomial distribution and tested for enrichment over the control sample with a Wald's Test. The window variance ϕ_k is assumed to be the same across all samples for the same window position k and is estimated by a kernel functions from all windows in the local region. Windows that pass the p-value cutoff are merged and optionally refined. For differential binding experiments, the normalized input reads are subtracted from the normalized ChIP reads for each window, and both directions are tested with one ChIP group being the test and the other being control.

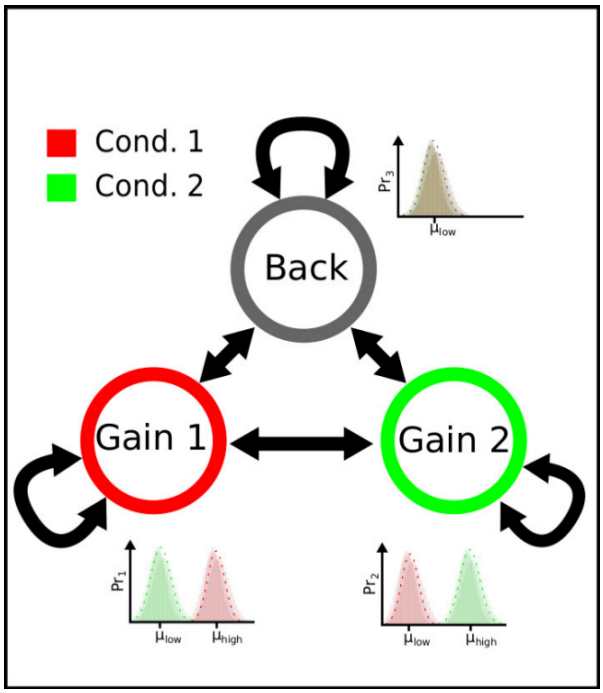

Figure 1.9: Hidden Markov Model for differential binding. Taken from THOR, Ref. [109], Figure 1A. Only the HMM part of Figure 1, penal A was taken. Shown is a three state HMM with one state for Background (grey), one state for condition 2 having a higher read count over condition 1 (green) and one state for condition 1 having a higher read count over condition 2 (red). The different read count distributions are schematically shown next to the states.

1.2.2 Methodological issues of state-of-the-art methods

Parts of the introduction presented in this section are part of the manuscript "GenoGAM: genome-wide generalized additive models for ChIP-Seq analysis" from Stricker and Engelhardt et al. 2017 [1].

Analysis of ChIP-Seq data is an essential part of a genomics pipeline that is performed on a regular basis. In order to identify the best methods for a given analysis, a number of benchmarks has been conducted with inconsistent performances for each inspected algorithm, leading to varying conclusions about the usability of each method [132, 133, 134, 135, 136, 137]. Despite the lack of clarity, there have been efforts to identify features that generally work in favor of an analysis [138, 139].

These efforts have shown a number of components for a better performance: First, differential occupancy tools achieve a higher precision if they can handle replicates. Thus most newer methods support biological replicates. These include diffReps [101], where a sliding window moves along the genome in a fixed step size and a robust test based on negative binomial distribution is performed on the number of reads falling into the window. PePr [99] follows a similar scanning approach and estimates local variance (see also Figure 1.8). THOR [109] uses a HMM approach to segment the genome into regions that are enriched, depleted or not differentially occupied (see also Figure 1.9). Complementary to testing for overall occupancies, MMDiff [94] allows testing for differences in shapes in given regions.

However, those methods lack the capability of comparisons between more than two groups of samples or supporting any full factorial designs including crossed designs. One way to handle different experimental designs is to use an approach based on generalized linear model (GLM) [140] as demonstrated by DESeq/DESeq2 [124, 141] and edgeR/csaw [125, 98]. DESeq [124] tests for differential overall occupancies at predefined regions of interest by testing for differences in number of reads overlapping the region. While csaw [98] devises how to test differential occupancies across windows in given regions using edgeR [125] while properly controlling for FDR.

Second, tunable parameters often enable model flexibility for a particular analysis at hand. For instance, allowing to test for different window sizes. Thus many methods provide a large set of tunable parameters which in turn requires extensive parameter fine tuning to achieve a good trade-off between precision and recall [139]. More importantly, it adds a substantial level of subjectivity to each analysis. By far the most important parameter to tune is window size, sometimes accompanied by step size. The latter indicates by how many basepairs the window moves along the genome. A step size of one is equivalent to a sliding window. A common default value is approximately 200bp or a multiple of estimated or given fragment size, which yields a similar number. The justification lies on the one hand in the selected fragment size for most library preparation protocols in ChIP-Seq, which is around 200bp [142]. On the other hand it is approximately the length of the DNA in a single nucleosome including wound and linker DNA, which is especially of interest in applications related to histone modifications [142]. While the latter might carry some biological meaning, the former is a technical byproduct

of the ChIP-Seq protocol, since TFBS range from 8 to 20bp [143], while histone marks express signals of various lengths [144]. An optimal window size will maximize useful information content and minimize the incorporation of noise, while being generalizable across a number of different profiles (see Figure 1.7b). This means, estimation of window sizes should be based on the underlying biology rather than the artificial resolution of ChIP-Seq. A more objective approach has been demonstrated by the peak caller JAMM [145], which minimizes a cost function that balances window size with the density of reads [146].

Another problematic area is the control of type I error through FDR. Its loss is demonstrated in Lun et al. [98] for differential binding and more recently in Chitpin et al. for peak calling [147]. Both examine the pitfalls of the common two step strategy that leads to data snooping. In a first step, the data is used to identify candidate regions and in the second, those regions are further examined for peaks (in peak calling) or analyzed for differential occupancy (in differential binding). In both steps the same data is used to identify and analyze the regions. Furthermore, this strategies can produce biased p-values (see Figure 1.10) leading to a loss of basis to select a proper threshold as well as loss of trust in the called peaks of differential regions. This is often mirrored in the different default thresholds across all methods requiring fine tuning for each method and analysis. Because of this, methods like JAMM [145] refrain from implementing a proper cutoff, but rather advise to use the Irreproducible Discovery Rate (IDR) [148]. It measures the consistency between replicates and uses reproducibility in score rankings between peaks in each replicate to determine an optimal cutoff for significance.

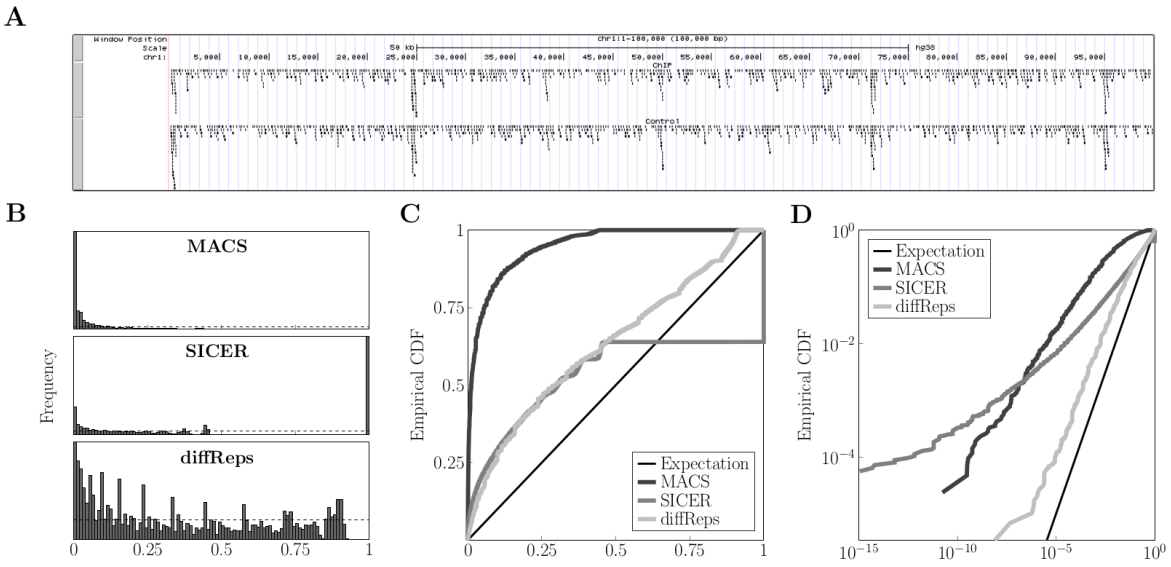

Figure 1.10: Biased p-values by MACS, SICER and diffReps. Taken from Ref. [147], Figure 1, panel A - D. **(A)** Visualization of part of a simulated ChIP-Seq read data set, with 500bp foreground regions every 20-25kbp, where read density is greater. Control data was generated similarly, with matching foreground regions, so a null hypothesis of no enrichment in IP versus control is true for every possible genomic region. **(B)** Peaks called by the three algorithms have p-values that are not uniformly distributed between zero and one, as should be the case for this null hypothesis data if p-values were well calibrated. **(C,D)** Empirical cumulative distribution functions on linear **(C)** and log **(D)** axes also show the discrepancy from the uniform distribution.

1.3 Scope of the Thesis

Parts of the introduction presented in this section are part of the manuscript "GenoGAM: genome-wide generalized additive models for ChIP-Seq analysis" from Stricker and Engelhardt et al. 2017 [1].

In my thesis I developed a new method called GenoGAM (genome-wide generalized additive models) which brings generalized additive models (GAM) to genomic applications. GAMs are extensions of GLMs for which covariates can be modeled as smooth functions [149]. I use them to model ChIP-Seq count rates along the genome. GenoGAM normalizes for sequencing depth and can handle factorial experimental designs, including any number of biological replicates and multiple controls. The amount of smoothing is estimated in an automatic, data-driven manner and thus avoids introducing subjectivity from the user. Moreover, well-calibrated per-base-pair p-values and region-wise p-values can be obtained to correctly control for type I error rates. In this thesis I will explore the statistical foundations and efficient implementation of GenoGAM, as well as a number of different applications and benchmarks:

First, the statistical and mathematical background of GenoGAM is given (Chapter2). The adaptation of the model to genomics, in particular differential binding, is described and benchmarked against state-of-the-art methods (Chapter 3). By re-analyzing data from a previous study on histone H3 Lysine 4 trimethylation (H3K4me3) [121] comparing wild type yeast versus a mutant, GenoGAM is shown to be more sensitive than current methods for testing differential occupancy, while still controlling for type I error rate.

Second, a number of algorithms are described that are adapted for the implementation of GenoGAM to ensure its scalability to applications for gigabase-scale genomes (Chapter 4). Runtime and memory efficiency are demonstrated and compared with the previous version of GenoGAM, which was based on a general implementation of GAMs. By computing whole-genome fits for human ChIP-Seq datasets in less than 9 hours on a standard 60-core server GenoGAMs utility is shown. On top of this, consistent biological accuracy is shown by replication of all previous ChIP-Seq studies from chapter 3 and chapter 5.

Finally, GenoGAMs flexibility is demonstrated by application to peak calling and a proof-of-concept modeling of bisulfite sequencing data using a quasi-binomial distribution (Chapter 5).

2 Statistical and mathematical background

Before the GenoGAM model can be described, an introduction to a few core components is needed. This chapter will give an overview to generalized additive models, which are at the heart of GenoGAM. Explain spline functions, which are responsible for smoothing the count data, with a focus on B- and P-splines. Finally, a very brief introduction is made into sparse matrices and why they are essential to solve the scaling problem of GenoGAM.

2.1 Generalized Additive Models

2.1.1 The model

A generalized additive model (GAM) [149] is a generalization of a generalized linear model (GLM), where covariates can be modeled as a sum of smooth functions additionally to the linear predictor $\mathbf{X}\boldsymbol{\theta}$. The model structure can be represented as follows:

$$g(\mu_i) = \mathbf{X}_i\boldsymbol{\theta} + \sum_k f_k(x_{ki}) \tag{2.1}$$

where g is a link function, $\mathbf{X}_i$ the i-th row of the design matrix $\mathbf{X}$, $\boldsymbol{\theta}$ the corresponding parameter vector, f_k the smooth functions of covariates x_k and $\mu_i = E(y_i)$, the expectation of y_i, which is a random variable assumed to be distributed according to a exponential family distribution. The dependence of response variable y on the covariates can thus be specified very flexibly in terms of any suitable function f. In fact f_k doesn't have to be the same for all k but can vary for each covariate.

In theory, a *suitable function* is any smooth function that can be represented as a linear combination. This can range from a simple running mean smoother, over a running lines smoother to kernel and spline smoothers. A more complete description of smoothers can be found in chapter 2 of Hastie and Tibshirani (1990) [150]. In practice, cubic splines are the preferred functions in GAMs. They are piecewise polynomials that can be defined as follows:

$$f(x) = \sum_{r=1}^{p} b_r(x)\beta_r \tag{2.2}$$

with $b_r(x)$ as a basis function and β_r its respective coefficient. Plugging equation (2.2) into equation (2.1) yields a GLM with $b_r(x)$ functioning as a covariate variable:

$$g(\mu_i) = \mathbf{X}_i\boldsymbol{\theta} + \sum_k \sum_{r=1}^{p} b_{rk}(x_{ki})\beta_{rk} \tag{2.3}$$

Therefore, the spline functions can be merged with the linear predictor into a simplified matrix-vector notation, yielding:

$$g(\mu_i) = \tilde{\mathbf{X}}_i\tilde{\boldsymbol{\beta}} \tag{2.4}$$

with $\tilde{\mathbf{X}} = (\mathbf{X}, b_1(x), b_2(x), \ldots, b_p(x))$ and $\tilde{\boldsymbol{\beta}}^T = (\boldsymbol{\theta}^T, \boldsymbol{\beta}^T)$.

For convenience the tilde over the variables will be dropped and the complete design matrix referred to as $\mathbf{X}$ and the parameter vector referred to as $\boldsymbol{\beta}$. Once the model is specified, the parameters guarding the relationship between response and covariates can be estimated. In the case that the link function g is just the identity function, for instance if $y_i \sim \mathcal{N}(\mu_i, \sigma)$, parameter estimation can be performed analytically by least square estimation just like in a regular linear model. However, if y_i is not normally distributed with a non-identity link function g, other approaches have to be employed to find a satisfactory set of parameters.

2.1.2 Parameter estimation

Least square estimation is ultimately concerned with finding the orthogonal projection of $\boldsymbol{y}$ on to the space of $\mathbf{X}$. The identity link keeps the model linear and allows to find a suitable linear combination of the columns of the design matrix by solving a linear system. Whereas a non-linear link function, like the log function, makes it impossible to find an exact orthogonal projection. Therefore, the idea is to approximate the solution by iteratively solving a weighted least square problem of residuals regressed on the response vector. Hence readjusting, or re-weighting the fit at each step till convergence. This is called Iteratively Reweighted Least Squares (IRLS) [140] and is the most common method used to estimate parameters in GLMs. The algorithm based on Wood (2017) [151] is provided below (see Algorithm 1). The derivation of the algorithm can be found in Wood (2017) [151] or in Nelder and Wedderburn (1972) [140].

In principle, IRLS is a Newton-Raphson method that is iteratively minimizing quadratic approximations of the objective function around the current estimate. Then, the current parameter vector $\boldsymbol{\beta}$ can be updated as:

$$\boldsymbol{\beta}_{t+1} = \boldsymbol{\beta}_t - \mathbf{H}^{-1}(\boldsymbol{\beta}_t)\nabla f(\boldsymbol{\beta}_t) \tag{2.5}$$

where the inverse Hessian $\mathbf{H}^{-1}(\boldsymbol{\beta}_t)$ captures the local curvature of the objective function, and the gradient vector $\nabla f(\boldsymbol{\beta}_t)$ captures the local slope. The iteration stops when the change in the log-likelihood or the norm of the gradient of the log-likelihood falls below a specified convergence threshold. Generally, for distributions from the exponential family, the penalized log-likelihood is convex and thus convergence is guaranteed.

However, IRLS differs from the default Newton-Raphson method in one core component: Instead of using the Hessian matrix it uses the Fisher information matrix

$\boldsymbol{\mathcal{I}}(\boldsymbol{\beta}_t) = -\mathrm{E}[\mathbf{H}(\boldsymbol{\beta}_t)]$, that is the negative expectation over the Hessian matrix. In this case the algorithm is also know as the *Fisher Scoring Algorithm*. If the canonical link function is used, both matrices are the same [152]. Otherwise they might differ. This is for example the case for the negative binomial distribution with known dispersion parameter θ (a different θ than used above). The canonical link in this case is $\ln(\frac{\mu}{\mu+k})$. However, often the more intuitive link $\ln(\mu)$ is used [153].

Algorithm 1 Iterative Reweighted Least Squares

while convergence criteria not met **do**
  $\eta_i^{[t]} \leftarrow \mathbf{X}_i \hat{\boldsymbol{\beta}}^{[t]}$
  $\mu_i^{[t]} \leftarrow g^{-1}(\eta_i^{[t]})$
  $z_i^{[t]} \leftarrow g'(\mu^{[t]})(y_i - \mu_i^{[t]}) + \eta_i^{[t]}$ ▷ calculate pseudodata $\boldsymbol{z}^{[t]}$
  $W_{ii}^{[t]} \leftarrow \frac{1}{V\mu_i^{[t]} g'(\mu_i^{[t]})^2}$ ▷ calculate weight matrix $\mathbf{W}^{[t]}$

  $\hat{\boldsymbol{\beta}}^{[t+1]} \leftarrow (\mathbf{X}^T\mathbf{W}^{[t]}\mathbf{X})^{-1}\mathbf{X}^T\mathbf{W}^{[t]}\boldsymbol{z}$ ▷ Solve to obtain $\hat{\boldsymbol{\beta}}^{[t+1]}$
  $t \leftarrow t + 1$ ▷ Increment t
end while

Once the final parameter vector $\hat{\boldsymbol{\beta}}$ is obtained, we can compute the vector of fits $\hat{\boldsymbol{\mu}}$. Since $\hat{\boldsymbol{\mu}}$ are also estimates, one is generally interested in their variance $\mathbf{V}_{\hat{\boldsymbol{\mu}}}$, so called *standard error*, often used to derive confidence intervals. Thus we have:

$$\mathbf{V}_{\hat{\boldsymbol{\mu}}} = \mathbf{X}\mathbf{V}_{\hat{\boldsymbol{\beta}}}\mathbf{X}^T = \mathbf{X}(\mathbf{X}^T\mathbf{W}\mathbf{X})^{-1}\mathbf{X}^T = \mathbf{X}\mathbf{H}^{-1}\mathbf{X}^T \tag{2.6}$$

with the standard errors σ_{ii} on the diagonal of the final matrix. In case of a normal distribution with identity link equation (2.6) simplifies to

$$\mathbf{V}_{\hat{\boldsymbol{\mu}}} = \mathbf{X}\mathbf{V}_{\hat{\boldsymbol{\beta}}}\mathbf{X}^T = \sigma^2\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T \tag{2.7}$$

A more complete introduction of GLMs and GAMs can be found in Hastie and Tibshirani (1990) [150] and Wood (2017) [151].

2.2 Penalized B-Spline functions

An essential step towards the specification of a generalized additive model is the selection of a spline function. In practice, there are a handful that are widely used, and their choice depends on the problem at hand. From equation (2.2) it can be seen that a spline function is made up of a set of basis functions $b_r(x)$. One straightforward way to define such a function could be as a regular polynomial, e.g. $b_1(x) = 1, b_2(x) = x, b_3(x) = x^2, \ldots, b_r(x) = x^{r-1}$. However, polynomials tend to have problems when it comes to model effects over an entire domain [151]. Thus, a more practical way are piecewise polynomials, where neighbouring pieces are joined at so called *knots*. Although theoretically each data point could be a knot, formally equivalent to a smoothing spline,

it will be computationally expensive. In particular for longitudinal data. Therefore a strategy is needed for the number and placement of knots.

One very widely used spline function basis is the B-spline basis [154], most easily defined recursively:

$$B_r^{-1}(x) = \begin{cases} 1 & x_r \leq x < x_{r+1} \\ 0 & \text{otherwise} \end{cases} \tag{2.8}$$

$$B_r^m(x) = \frac{x - x_r}{x_{r+m+1} - x_r} B_r^{m-1}(x) + \frac{x_{r+m+2} - x}{x_{r+m+2} - x_{r+1}} B_{r+1}^{m-1}(x), \quad r = 1, \ldots, p$$

with $m + 1$ the order of the basis function, p the number of parameters and each x_r a knot. In general, the order of the basis function should be at least cubic (here $m = 2$) in order to allow for twice differentiation. Moreover, as can be seen from the indices of x, a p parameter B-spline requires $r + m + 2$ knots. That is, for a cubic spline four more knots than parameters. Another characteristic that can be seen from equation (2.8), is the local definition of B-splines bases over a finite support and zero otherwise. This is an advantage when it comes to smooth over subsequent positions, where effects of a given position should be kept local, not affecting other distant positions. Figure 2.1 shows an example cubic spline with 10 B-spline basis functions (the bell shaped curves) in the space $[0, 1]$. Six of the ten knots can be seen (black dots at the bottom), while the other four (two on each side) are outside of the evaluation space. However their function curves can be partially seen. The bold blue smooth above the curves is the complete spline, where at each position the value of the single curves were added.

Due to its properties, the B-spline basis is very flexible and capable of representing any possible spline function through a linear combination. This flexibility comes with a downside: As the positioning and total number of the knots are left to the user, a strategy has to be found to avoid over- or undersmoothing. In 1996 a possible solution was proposed by Eilers and Marx, introducing the concept of penalized B-splines (P-splines) [155]: If a sufficiently high number of knots are placed, the spline coefficients can be penalized through second order differences, which function as an approximation of second order derivatives:

$$l(\boldsymbol{\beta}, \boldsymbol{y}) - \lambda \sum_{r=l+1}^{p} (\Delta^l \beta_r)^2 = l(\boldsymbol{\beta}, \boldsymbol{y}) - \lambda \boldsymbol{\beta}^T \mathbf{S} \boldsymbol{\beta} \tag{2.9}$$

with $l(\boldsymbol{y}, \boldsymbol{\beta})$ the log-likelihood of the model and $(\Delta^l \beta_r)^2$ the difference penalty of order l, which is commonly set to $l = 2$. The penalization is controlled by a smoothing parameter λ that is estimated from the data by Generalized Cross Validation (GCV). Equation (2.9) can be rewritten in matrix-vector notation, such that $\mathbf{S}$ represents a symmetric positive matrix that encodes the squared second order differences of parameter vector $\boldsymbol{\beta}$. Figure 2.2 illustrates the concept. Panel A shows the wiggly smooth due to a high number of knots prior to penalization. Panel B shows the same fit after penalization has been applied according to equation (2.9). A more complete overview of splines can be found in deBoor (1978) [154] and Wood (2017) [151].

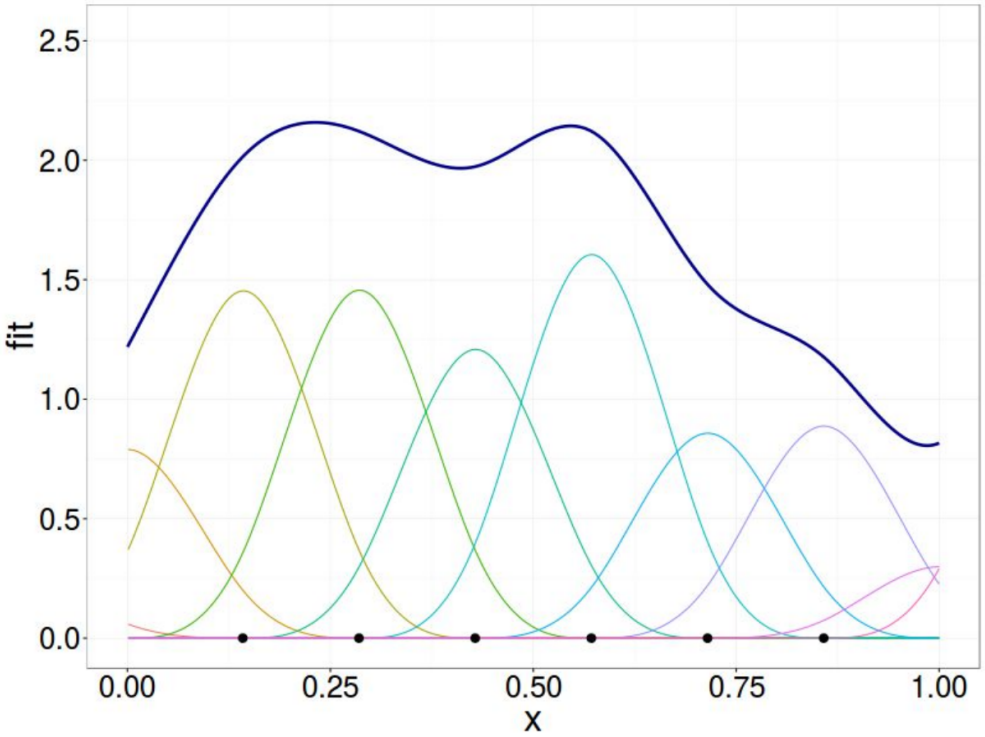

Figure 2.1: The concept of a B-spline function. Shown is an example cubic B-spline function with 10 basis functions (the colorful bell shaped curves) in the space $[0, 1]$. Six of the ten knots can be seen at the bottom of the x-axis (black dots), while the other four (two on each side) are outside of the evaluation space. However, parts of their function curves can be seen. Without the knots outside the space, the B-spline function would always go down to zero at the borders. The bold blue smooth above the curves is the complete spline, where at each position the value of the single curves were summed up.

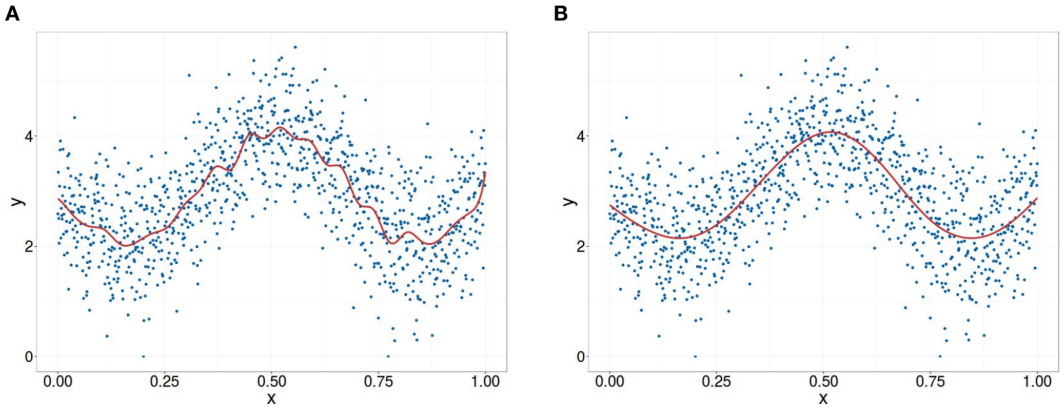

Figure 2.2: The concept of penalization of a B-spline function. (**A**) Due to a high number of knots within the space $[0, 1]$ the fit is wiggly. (**B**) The same fit from **A** is shown after penalization. The curve resembles a smooth fit as one would expect by eye from looking at raw data.

2.3 Sparse matrices

Sparse matrices constitute an important subset of matrices that are useful in solving computationally expensive linear systems. They are simply matrices with a large number of zeros. Although mathematically zero is a special number, computationally (that is, in an implementation) it is a normal number, that occupies storage space according to its type (e.g integer or double). Thus, in order to take advantage of sparse matrices a different storage approach is needed.

Intuitively, a straightforward implementation could store the elements in the so called *triplet* format. That is, in three vectors with column and row indices and their values, respectively. Although this is easy to generate, it is harder to use in most sparse direct methods [156]. Therefore, the inner representation of the sparse matrix object often makes use of the *compressed-column* form (and in case of the transpose the *compressed-row* form). An example based on an example from Davis (2006) [156] is shown in equation (2.10). C code notation is used below to illustrate the inner representation.

$$A = \begin{bmatrix} 1.8 & 0 & 2.7 & 0 \\ 3.0 & 0.8 & 0 & 1.3 \\ 0 & 7.4 & 4.7 & 0 \\ 2.1 & 5.5 & 0 & 0.2 \end{bmatrix} \tag{2.10}$$

```
int p [ ]    = { 0,              3,              6,        8,       10 } ;
int i [ ]    = { 0,   1,   3,   1,   2,   3,   0,   2,   1,   3   } ;
double a [ ] = { 1.8, 3.0, 2.1, 0.8, 7.4, 5.5, 2.7, 4.7, 1.3, 0.2 } ;
```

It shows a $n \times m$ matrix with $k <= nm$ non-zero values. Then the vector `i`, containing the row indices of each non-zero entry, and the vector `a`, containing the values, are of length k (here $k = 10$). The vector `p` is of length $m + 1$ and contains values that help map non-zero entries into the index space of `i` and `a`. That is, for a given column j the row indices and the non-zero elements are located in `i[p[j]]` through `i[p[j+1]-1]` or in `a[p[j]]` through `a[p[j+1]-1]`, respectively. Because `p` helps identify the rows for a given column, access to columns is very fast, while access to rows is expensive. Moreover modification of this structure, e.g. inserting a new non-zero value, is not trivial. By convention values that turn zero through computation are kept in as *non-zero zeroes*.

Methods developed to deal with sparse matrices take advantage of the structure to perform usual tasks like matrix multiplication, transpose computation or factorization faster and more memory efficient. A task that is often the computational bottleneck in fitting of statistical models is matrix inversion (see Algorithm 1 and equation 2.6). Although it can be sometimes solved without an actual matrix inversion by a direct solver (e.g. in Algorithm 1), this is not always possible (e.g. in equation 2.6). Matrix inversion can be split into three steps: *Permutation*, to find a more favorable representation. *Factorization* into a product of matrices to reduce the amount of computational steps and finally the solution of a linear system involving the factorized matrices. Almost all methods for sparse matrices split the factorization further into two stages:

1. *Symbolic factorization* computes an ordering, often using nested dissection or a approximate minimum degree algorithm, such that the factors will be as sparse as possible and allocates space to hold the result. The value of the matrix entries are either not used or only used to make estimates about pivoting.

2. *Numeric factorization* computes the factorization given the ordering and sparsity computed by symbolic factorization.

Additionally, using supernodal, frontal, or multifrontal methods during symbolic factorization this strategy results in an efficient scheme of iterative operations on dense submatrices during numeric factorization. A more complete overview and detailed description of sparse matrix methods can be found in Davis (2006)[156].

Given a GAM without the parametric part and with the functions represented as B-splines, one can obtain a design matrix where each column is composed of non-zero values stemming from the respective basis function and a set of zeros which lie outside of the respective basis function. This does not necessarily guarantee a sparse matrix. However in modeling problems with increasing number of basis functions the non-zero set shrinks compared to the zero set (Fig. 2.3). In this cases sparse matrix methods can provide a significant boost in computation.

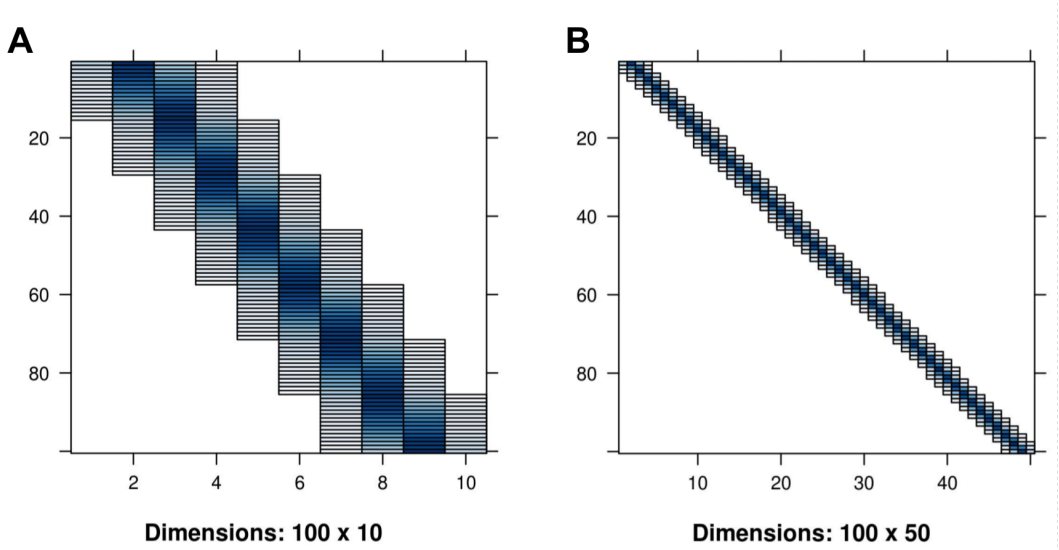

Figure 2.3: Example sparse B-spline design matrices. (**A**) A sparse matrix with 100 data points (rows) and 10 basis functions (columns). The white colored space are zeroes, whereas the blue scale represents non-zero values from low (gray) to high (blue). Note, that the matrix is not squared, it is deliberately plotted this way for a better visualization. (**B**) A sparse matrix with 100 data points (rows) and 50 basis functions (columns). The white colored space are zeroes, wheres as the blue scale represents non-zero values from low (gray) to high (blue). Due to an increase in basis functions the zero set has increased relatively to the non-zero set. Note, that the matrix is not squared, it is deliberately plotted this way for a better visualization.

3 GenoGAM model for differential occupancy

The methodology, results and figures presented in this section are part of the manuscript "GenoGAM: genome-wide generalized additive models for ChIP-Seq analysis" from Stricker and Engelhardt et al. 2017 [1].

3.1 A generalized additive model for ChIP-Seq data

Given an experiment consisting of a set of ChIP-Seq samples, a data point is defined by a pair of a ChIP-Seq sample and a genomic position. Denote by x_i the genomic position of the i-th data point, by j_i its ChIP-Seq sample and by $y_i \geq 0$ the number of fragments in sample j_i centered at position x_i. For paired-end libraries, the fragment center is calculated by shifting the read start or end position by a half of the fragment size. In case of single end data, the fragment length d is estimated using the Bioconductor package `chipseq` and its *coverage* method. It is defined as the optimal shift for which the number of bases covered by any read is minimized. Thus, the center was taken as the start of the read shifted by $\frac{d}{2}$ downstream. When reducing ChIP-Seq data to fragment centers rather than full base coverage, each fragment is counted only once. This reduces artificial correlation between adjacent nucleotides. The counts y_i are modelled using the following generalized additive model:

$$y_i \sim \mathrm{NB}(\mu_i, \theta) \tag{3.1}$$

$$\log(\mu_i) = o_i + \sum_{k=1}^{K} f_k(x_i)\, z_{j_i,k} \tag{3.2}$$

The counts y_i are assumed to follow a negative binomial distribution with means μ_i (Equation 3.1) and a dispersion parameter θ that relates the variance to the mean such that $\mathrm{Var}(y_i) = \mu_i + \mu_i^2/\theta$. Consequently, the model accounts for dispersion beyond Poisson noise [124].

The logarithm of the mean μ_i is the sum of an offset o_i and one or more smooth functions f_k (Equation 3.2). The offsets o_i are predefined data-point specific constants that account for sequencing depth variations (see section 3.1.3). The indicator variable $z_{j_i,k}$ is 1 if the smooth function f_k contributes to the mean counts of sample j_i and 0 otherwise (see Table 3.1 for a schematic representation). As shown below, this formulation allows modeling IP versus input experiments as well as factorial experimental designs.

	$f_1(x)$	$f_2(x)$	...	$f_K(x)$
sample 1	1	1	...	0
sample 2	0	1	...	0
...	...	...	...	...
sample J	0	0	...	1

Table 3.1: Schematic factorial design table of variable $z_{j_i,k}$

In the following the experimental setting of IP versus input will be used to explain the fundamental components and functionalities of GenoGAM.

IP versus input experiments are modelled using GenoGAM with two smooth functions: f_{input} that contributes to both input and IP samples, and f_{protein} that only contributes to IP samples. More specifically, f_{input} models local ChIP-Seq biases common to input and IP, whereas f_{protein} models the protein log-occupancy up to one genome-wide scaling factor. Figure 3.1 shows the application of this model to one ChIP-Seq library for the *S. cerevisiae* general transcription factor TFIIB and its input control (see Appendix A.3.1 for the experiment protocol).

In GenoGAM, the smooth functions are represented by cubic spline curves, which are written as linear combinations of a set of regularly spaced basis functions b_r, i.e. $f_k(x) = \sum_r \beta_r b_r(x)$. Second order B-splines are chosen as basis functions, which are bell-shaped cubic polynomials over a finite support [154] (see section 2.2). To avoid overfitting, regularization of the functions f_k is carried out by penalization of the second order differences of the spline coefficients, which approximately penalizes second order derivatives of f_k – an approach called P-Spline or penalized B-splines [155] (see section 2.2). The optimization criterion for these P-Splines is the sum of the negative binomial log-likelihood (depending on the response vector $\boldsymbol{y}$ and the vector $\boldsymbol{\beta}$ containing the coefficients of all smooth functions) plus a penalty function that is weighted by the smoothing parameter λ (see also Equation 2.9):

$$\hat{\boldsymbol{\beta}} = \text{argmax}\{\ l_{\text{NB}}(\boldsymbol{\beta}; \boldsymbol{y}, \theta) - \lambda \boldsymbol{\beta}^T (\mathbf{S} + \epsilon \mathbf{I}) \boldsymbol{\beta}\} \tag{3.3}$$

where the $\epsilon\mathbf{I}$ term adds regularization on the squared values of the $\boldsymbol{\beta}$s, which is particularly useful for regions with many zero counts. This regularization allows dense placements of the basis functions (between 20 and 50 bp), while relying on the smoothing parameter λ to protect against overfitting. Large values of λ yield smoother functions. A single smoothing parameter common to all smooth functions proved to be sufficient for the applications. For given λ and θ, model fitting can be performed by Penalized Iteratively Reweighted Least Squares (P-IRLS) (see section 2.1.2 and following section 3.1.1).

Adapting a Bayesian view, the penalized likelihood can be interpreted as a posterior probability, and the penalization term arises from a Gaussian prior on the parameters $\boldsymbol{\beta}$. Large-sample approximations then yield a multivariate Gaussian posterior distribution

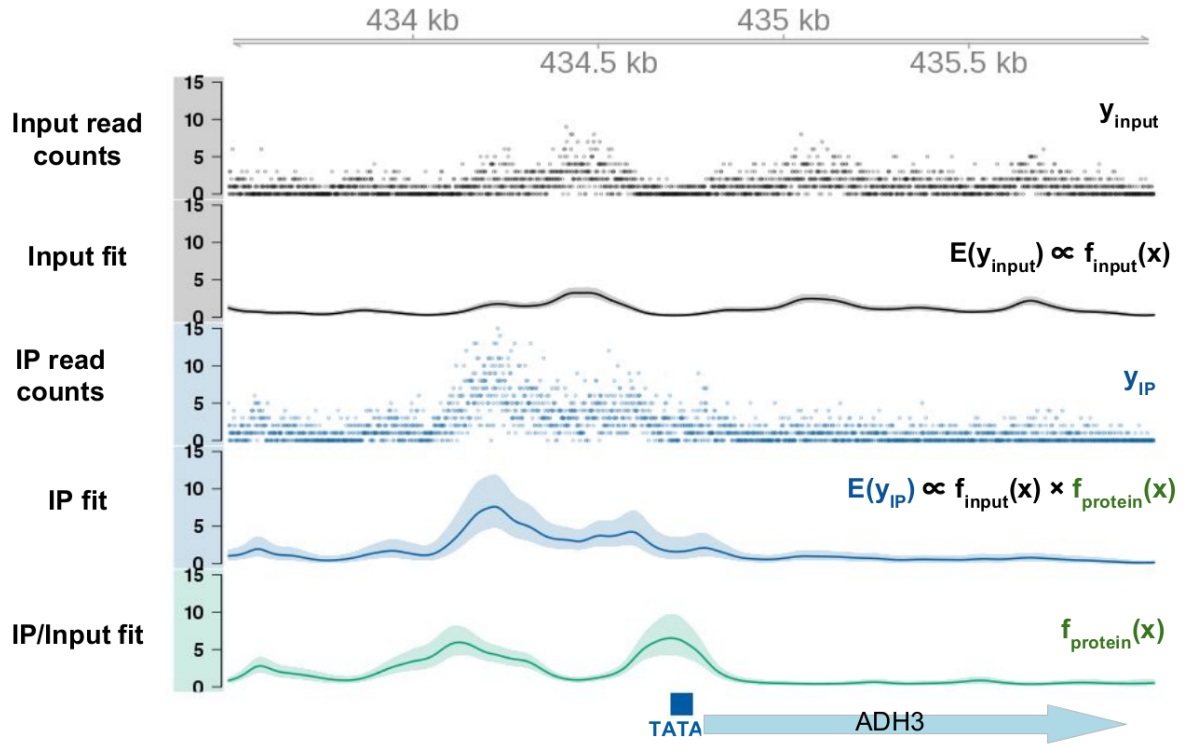

Figure 3.1: GenoGAM model overview ChIP-Seq analysis with GenoGAM yields base-pair resolution occupancy profiles with confidence bands. Input (black) and IP (blue) centered read counts (dots) and fitted smooth (solid line) with 95% confidence intervals (ribbons) for the transcription factor TFIIB for a section of the chromosome XIII of *S. cerevisiae*. Additionally, the extracted fold change of IP over Input (green) and gene annotation at the very bottom. Simplified equations depict model constituents.

for $\boldsymbol{\beta}$, and, by the linearity of $f_k(x) = \sum_r \beta_r b_r(x)$, Gaussian posteriors for the point estimates $f_k(x)$. This allows for the construction of pointwise confidence bands [151]. An example of the fitted smooth functions and their confidence bands for the yeast transcription factor TFIIB is shown in Figure 3.1.

3.1.1 Fitting of a GAM on a genome-wide scale, given the smoothing and dispersion parameters λ and θ

Since the computation time of a GAM grows polynomially with the number of basis functions, fitting one big model might be unfeasible. Instead, GAMs are fitted separate on sequential overlapping intervals (or tiles, Fig. 3.2).

Each chromosome is partitioned into equally-sized intervals called *chunks*. *Tiles* are defined as chunks extended on either side by equally-sized *overhangs*. The generalized additive model is fitted on each tile separately and point estimates and their standard

errors extracted at each base pair of the smooth functions. The tile fits are then restricted to their chunk to define the chromosome-wide fit.

As overlap length increases, agreement of the fit at the midpoint of the overlapping region increases. A genome-wide fit is obtained by joining together tile fits at overlap midpoints (Fig. 3.2). This approximation yields computation times that are linear in the number of basis functions at no practical precision cost (Fig. 3.3). Furthermore, it allows for parallelization, with speed-ups being linear in the number of cores (Fig. 3.4). This approximation parallelizes the computation over the data, which might allow future implementation of GenoGAM in map-reduce frameworks such as Spark [157].

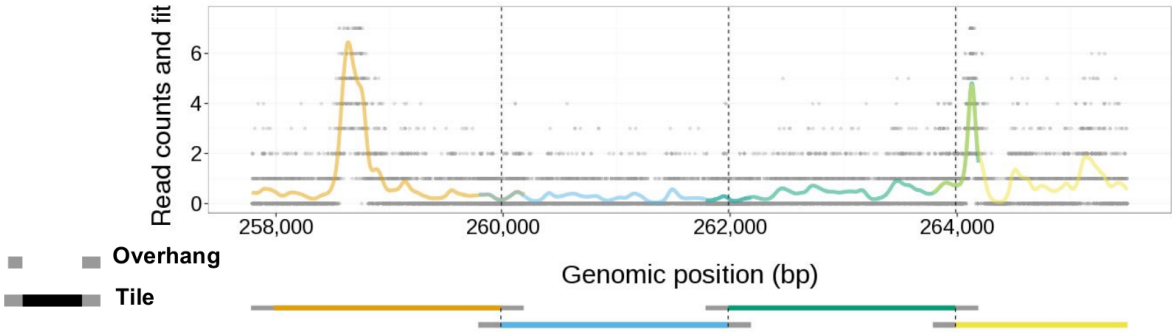

Figure 3.2: GenoGAM splits data into overlapping chunks (tiles) Read count (black dots, capped at ≥ 7) and predicted rates (orange, blue, green, and yellow transparent lines) for four successive tiles (lower track). Vertical dashed lines denote the junction points

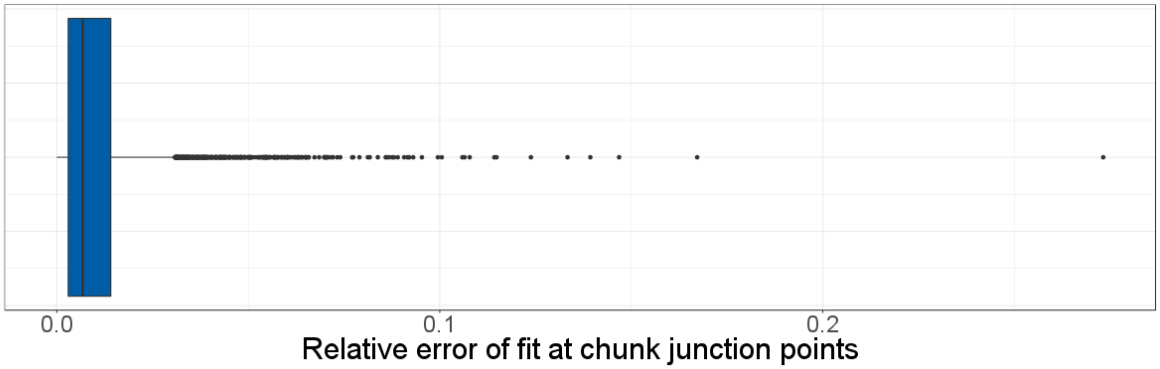

Figure 3.3: Relative error distribution at junction positions Distribution of the relative error (difference over mean) at the junction point of neighboring tiles, for an overhang of 8 basis functions.

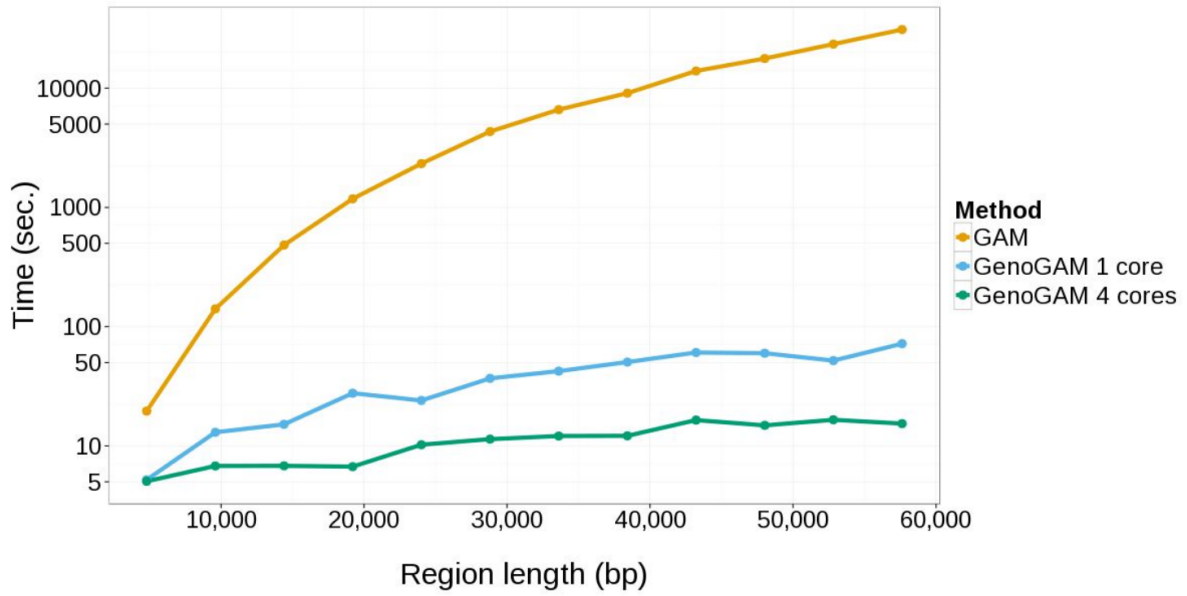

Figure 3.4: Embarrassing parallelization decreases computational runtime Computing time in seconds (y-axis in log scale) versus region length in bp (x-axis) for a standard GAM (orange), GenoGAM on a single core (blue), and GenoGAM on four cores (green). Tiles were 2,400 bp long and contained 100 basis functions each. All fits were performed with `mgcv`.

3.1.2 Data-driven determination of the smoothing and dispersion parameters λ and θ

To determine the optimal value for λ and θ, generalized cross-validation was tried, an analytical leave-one-out large-sample approximation [151]. However, this yielded very wiggly fits indicative of overfitting. Thus, an empirical cross-validation scheme was developed.

For efficiency, cross-validation is performed using only a subset of the data. A sufficiently large set of distinct regions is selected that are long enough to not suffer from border effects common to spline fitting. Using 20 or more distinct regions containing at least 100 basis functions gave satisfactory empirical results. Regions are selected that have the most significant fold change of IP versus input read counts.

In each region, 10-fold cross-validation is performed, where a tenth of the data points are removed, the model is fitted on the remaining data points, and the log-likelihood of the left-out data points is computed. To avoid overfitting due to short range correlations, each cross-validation fold does not consist of randomly selected single genomic positions, which would recapitulate the leave-one-out scheme, but of short intervals. The length of these intervals is set to 20 bp (approximately a tenth of the fragment length) in absence of replicates and twice the average fragment sizes when replicates are available.

For a given pair of values for λ and θ, the score function is defined as the sum of out-of-sample log-likelihood over all cross-validation folds and all tiles, restricted to the data

points within chunks to not depend on poor fitting in overhangs. Investigation on grid values of θ and λ showed that the out-of-sample log-likelihood was typically unimodal. Therefore the Nelder-Mead numerical optimizer is used to jointly fit the two parameters [158].

3.1.3 Sequencing depth variations

In estimating sequencing depth variations, an approach is used originally suggested by Meyer (2014) [159], that is robust to variations in signal-to-noise ratio. Variations for sequencing depth is controlled by using size factors computed by DESeq2 ([141]). This method robustly estimates fold-changes in overall sequencing depth by comparing read counts of predefined regions. The selection criteria for these regions is application-specific. For differential binding application, all tiles are considered. Whereas for peak calling a set of the top tiles by fold change might proof for a better strategy.

3.2 Differential binding

3.2.1 GenoGAM model

In an differential binding application comparing mutant with wildtype ChIP data, the general model from section 3.1 is further specified. The following GenoGAM model is used:

$$y_i \sim \mathrm{NB}(\mu_i, \theta) \tag{3.4}$$

$$\log(\mu_i) = \log(s_{j_i}) + f_{\mathrm{WT}}(x_i) + f_{\mathrm{mutant/WT}}(x_i) z_{j_i,\mathrm{mutant}} \tag{3.5}$$

where $z_{j_i,\mathrm{mutant}} = 1$ for j index of mutant samples and 0 for wild-type samples. The offsets $\log(s_{j_i})$ are log-size factors computed to control for sequencing depth variation and overall H3K4me3 across all four samples (see section 3.3 for experimental setting). Table 3.2 illustrates the factorial design.

	$f_{\mathrm{WT}}(x)$	$f_{\mathrm{mutant/WT}}(x)$
WT_1	1	0
WT_2	1	0
Mutant_1	1	1
Mutant_2	1	1

Table 3.2: Factorial design table for the differential binding application with two smooth functions and four samples in total (two replicates each for wildtype and mutant)

3.2.2 Position-level significance testing

Null hypotheses of the form $H_0 : f_k(x) = 0$ for a smooth function f_k at a given position x of interest are tested assuming approximate normal distribution of the corresponding z-score, i.e.:

$$T_k(x) = \frac{\hat{f}_k(x)}{\hat{\sigma}^2_{f_k(x)}} \sim N(0,1) \tag{3.6}$$

where $\hat{f}_k(x)$ and $\hat{\sigma}^2_{f_k(x)}$ denote point estimate and standard error of the smoothed value.

3.2.3 False discovery rate for predefined regions

Let $R_1, ..., R_p$ be p regions of interest, where a region is defined as a set of genomic positions. Regions are typically, but not necessarily, intervals (e.g. genes or promoters). For instance, all exons of a gene could make up a single region. Regions can be a priori defined or defined on the data using independent filtering [160]. For instance, when testing for significant differences between two conditions, regions can be selected for having a large total number of reads over the two conditions [98].

For j in $1, .., p$, let H_0^j be the composite null hypothesis that the smooth function f_k values 0 at every position of the region R_j. The False Discovery Rate (FDR) is controlled as in Lun and Smyth (2014) [98]:

1. Position-level p-values at all region positions are computed using position-level significant testing as described above.

2. Within each region R_j, position-level p-values are corrected for multiple testing using Hochberg family-wise error rate correction [161]. The p-value for the null hypothesis H_0^j is then computed as the minimal family-wise error rate corrected position-level p-value. This step gives one p-value per region.

3. FDRs are controlled using the Benjamini-Hochberg procedure [162] applied to the region-level p-values.

As a concrete example take the bottom smooth track in figure 3.5. Given the coordinates of the gene *YNL176C*, at each position p-values are extracted (step 1). Those are corrected using the Hochberg family-wise error rate correction, obtaining a single p-value for this gene (step 2). Finally, a FDR is computed (step 3).

3.3 Benchmarking and results

To assess the performance of GenoGAM for calling differential occupancy, histone H3 Lysine 4 trimethylation (H3K4me3) ChIP-Seq data of a study [121] was re-analyzed comparing wild type yeast versus a mutant with a truncated form of Set1, the H3 Lysine 4 methylase. H3K4me3 is a hallmark of promoters of actively transcribed genes.

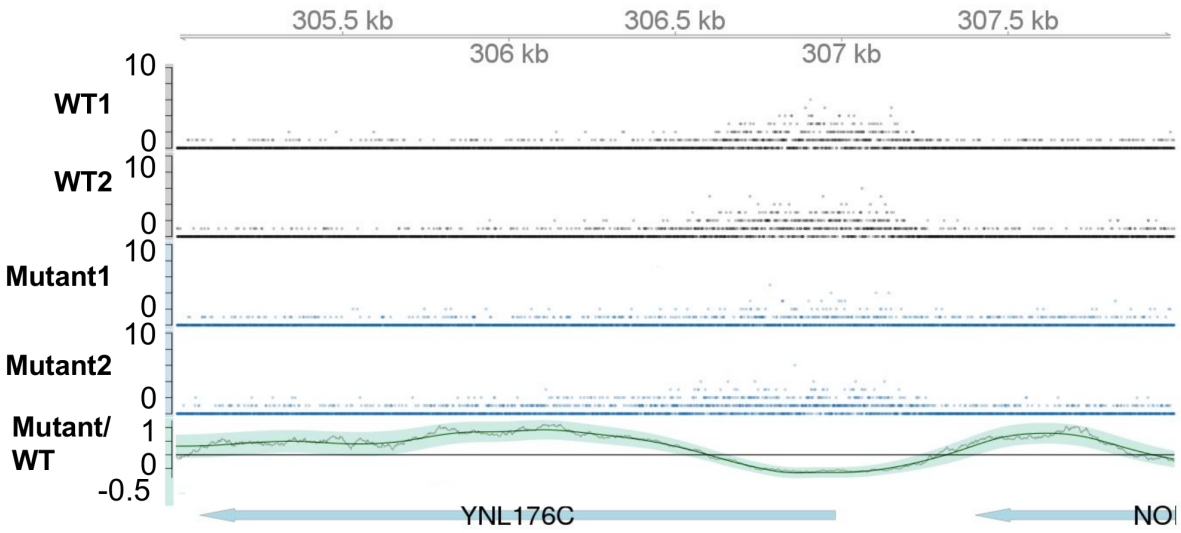

Figure 3.5: Differential occupancy model fitting Read counts (dots) and fitted rates with 95% confidence bands for wild-type (black) and mutant (blue) and the log-ratio of mutant over wild-type with confidence band (bottom row, green) around YNL176C. For comparison, log-ratios computed in sliding windows of size 184bp (bottom row, gray, optimized window size, see section 3.3.3.2)

Thornton and colleagues [121] have reported genome-wide redistribution in the truncated Set1 mutant of H3K4me3, which is depleted at the promoter and enriched in the gene body. This dataset is interesting for differential occupancy analysis because it is not about the overall number of counts, but about the redistribution of H3K4me3 within the gene. Hence, methods must be sensitive to differences at any location within the gene. We expect such redistribution of the mark at all genes that are transcriptionally active for yeast cells grown in rich media.

The two replicate IPs for mutant and for the wild type each were modeled with GenoGAM using one smooth function f_{WT} for the wild type reference occupancy, and one further smooth function $f_{\text{mutant/WT}}$ for the differential effect (see section 3.2.1). The offsets were computed to control for variations in sequencing depth between replicates and overall genome-wide H3K4me3 level (section 3.1.3). This yielded base-level log-ratio estimates and their 95% confidence bands genome-wide. Figure 3.5 shows an example region for data and fit at the gene *YNL176C* consistent with the report of reduced binding at promoter regions.

3.3.1 P-value calibration

Confidence bands of GAMs are formally Bayesian credible intervals. However, previous studies based on simulated data showed that these confidence bands have close to nominal coverage probabilities and can, in practice, be used in place of frequentist confidence intervals [163]. Base-level p-values are therefore estimated using the point-wise estimates and standard deviations according to equation 3.6. To empirically verify that

the p-values were at least conservative, a negative control dataset was created by per-base-pair independent permutation of the counts between the four samples. The offsets were set to 0 and the smoothing and dispersion parameters were estimated again. This non-parametric permutation scheme makes less assumptions than previous simulation studies [163].

Nonetheless, per-base-pair p-values in this negative control experiment were slightly overestimated (Fig. 3.6A). These results show that GenoGAM can be used to identify individual positions of significant differential occupancies with controlled type I error. Here, correction for multiple testing can either be done using the Benjamini and Hochberg procedure [162] (see section 3.2.3 above) or procedures that exploit dependencies between adjacent positions [164].

3.3.2 Competitor methods

Benchmarking was conducted against several state-of-the-art differential occupancy methods that proved to be competitive in a more recent benchmark [139]. Two more methods highlighted by Steinhauser et al. [139] were excluded: `ChIPComp`, as the `R` package is hardcoded to be used on mouse and human datasets only, and the `R` package `DiffBind`, which is redundant, since it is essentially a test for differences in overall counts based on either DESeq2 (already present) or edgeR (used by csaw). Furthermore the more recently published HMM-based method THOR [109] was included increasing the total number of methods to six. MMDiff [94] was used in the more recent second version (MMDiff2):

- `csaw` [98]
- `DESeq2` [141]
- `diffReps` [101]
- `MMDiff2` [94]
- `PePr` [99]
- `THOR` [109]

First, significance values were checked for proper calibration according to section 3.3.1 above. Since none of the methods computes positionwise p-values, calibration was checked by calling differentially bound regions at a nominal p-value of 0.05. If values are properly calibrated methods should return at most 5% of the regions being significant.

The R/Bioconductor packages `DESeq2` [141], `MMDiff2` [94], and `csaw`[98] were applied on the base-level permuted dataset for all genes. The log-size factors were set to 0 for all methods when applied to the permuted datasets. While DESeq2 allows for a count matrix as direct input, it was run on our permuted dataset. MMDiff2 and csaw on the other hand have their own read in functions in order to generate their initial object.

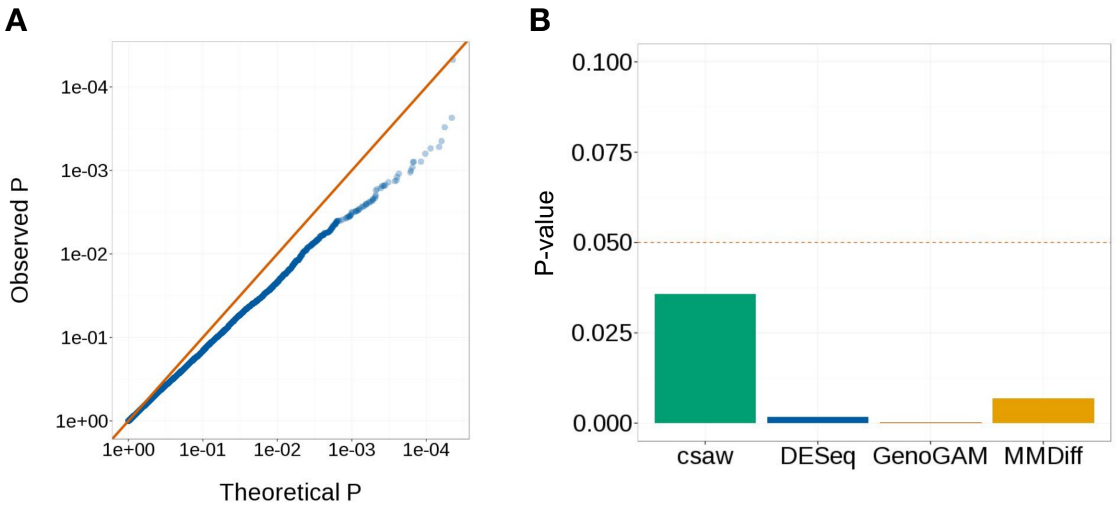

Figure 3.6: P-value calibration of GenoGAM and competitor methods (A) Empirical (y-axis) versus theoretical (x-axis) p-values in base-level permuted count data. P-values at every 200 bp positions are shown. **(B)** Proportion number of called genes on permuted data (false positives) at nominal p-value 0.05 (red dashed line).

Permutation of data was then performed within this object. For `THOR` [109], `PePr` [99] and `diffReps` [101] permutation of the data was not possible, as they directly operate from Binary Alignment Map (BAM) files and don't allow for easy interference in contrast to the above methods, which are implemented as an `R` package. Figure 3.6B shows the proportions of called genes for a given p-value of 0.05. All methods fall well below the 0.05 threshold, signaling conservative p-values.

On the original dataset DESeq2 was applied with default parameters. MMDiff2 was applied with a bin length of 20 bp, the center positions of the fragment and the MMD histogram distance. The csaw method was applied with window size of 150 bp and otherwise default parameters. The window size was determined through a grid search (see Figure 3.7), choosing the window size with the most significant genes. The reasoning is, that if most of the genes are transcribed (and thus most of them should come out significant) and the p-value is conservatively calibrated, then a higher number of called genes indicates a better performance. In particular, csaw uses a different procedure to estimate normalization factors than DESeq2. The default was used as it was in favor of csaw for returning more significant genes.

THOR, PePr and diffReps do not allow for an input of fixed regions, although THOR provides functionality to restrict the search to a set of pre-defined regions, which however yield worse results. Therefore differential binding sites were called and then overlapped with the gene coordinates. Any overlap of at least one base was counted as a positive call and thus the gene was assigned the respective significance value (the lowest if multiple overlaps were present). An alternative approach of using pre-filtered BAM/BED files as input yielded worse results. THOR was applied without any optional fields. That is,

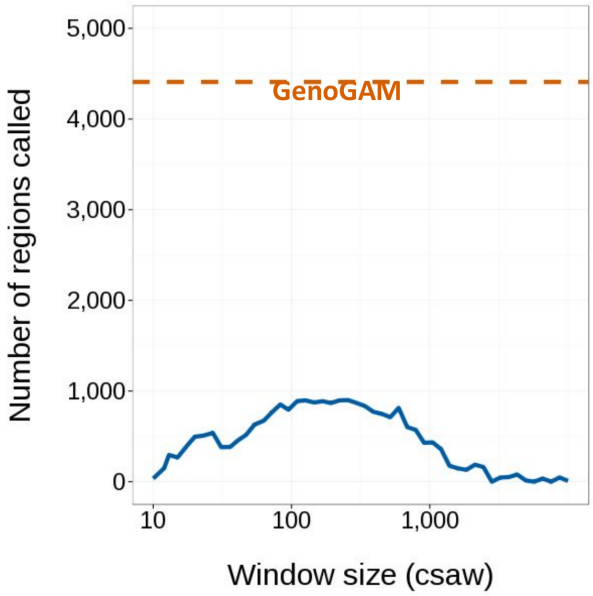

Figure 3.7: Number of significant genes by window size Number of genes with significant differential occupancies in mutant over wild type (FDR < 0.1) reported by csaw (blue) as function of window size (x-axis). For comparison, the dashed orange line marks GenoGAMs performance.

without GC-correction and input control and otherwise with default settings (binsize = 100, stepsize = 50 and p-value cutoff = 0.1). PePr was applied with default parameters, that is, an empirical estimation of fragment size and window size, a default p-value cutoff of 10^{-5} and the inter-sample normalization method. For diffReps also default parameters were used: window size of 1000bp, step size of 100bp and p-value cutoff of 0.0001.

3.3.3 Results

3.3.3.1 Higher sensitivity in testing for differential occupancy

First, GenoGAM was compared to csaw, which is its most directly comparable method because only GenoGAM and csaw can model flexible factorial designs and assess differences in overall read counts and in shape. One fundamental difference is that csaw is based on a sliding window approach requiring an a priori defined window size. In contrast, the smoothing parameter of GenoGAM is learnt from the data by maximizing the out-of-sample likelihood in cross-validation (see section 3.1.2). Across all investigated window sizes, the csaw algorithm reported a maximum of 863 significant genes at FDR < 0.1 (Fig. 3.7 and 3.8). Moreover, the number of identified genes depended strongly on the choice of the window size (Fig. 3.7). In contrast, GenoGAM reported 4,717 significant genes at the same FDR cutoff, which is much closer to the number of transcriptionally active genes [165]. The genes reported by GenoGAM included all the genes reported by csaw except two, indicating that GenoGAM captured the same signal

but with a higher sensitivity. The genes reported only by GenoGAM showed a differential occupancy pattern similar yet weaker to the genes common to csaw and GenoGAM, with depletion in the promoter and enrichment in the gene body (Fig. 3.8), indicating that GenoGAM captured true biological signal.

Next, GenoGAM was compared against the other occupancy methods. The least number of significant genes (FDR < 0.1)or the respective default threshold set by the method) were identified by DESeq2 (735), csaw (863) and diffReps (1193). The most were reported by THOR (2687), PePr (3248) and MMDiff2 (3482), closer to GenoGAM. The respective heatmap figures for all methods can be found in Appendix B.1.

To make sure that i) the reported genes indeed corresponded to transcriptionnaly active genes (Fig. 3.9) and ii) that these results did not depend on FDR cutoffs, receiver operating characteristic (ROC) were performed using expressed genes as a proxy for true positives. Gene expression levels were computed as the median normalized probe levels for the three replicate YPD conditions of all tiling array probes provided by Xu et al. (2009) [165] overlapping gene coordinates defined by Xu et al. (2009) [165]. Genes from Xu et al. (2009) [165] and from Thornton et al. (2014) [121] were matched by symbol.

To compute ROC curves, binary labels (expressed = 1 if above a given expression level quantile cutoff, or otherwise = 0) were assigned to each gene, and for each method, genes were ranked according to their respective significance value. For THOR, PePr and diffReps, genes that did not overlap any differentially bound site, p-values were set to 1. Then, ROC curves and AUC for all expression level quantile cutoffs in steps of 0.01 were computed.

GenoGAM had the largest area under the ROC curve (AUC), when considering that the 15% of the genes with lowest expression levels in Xu et al. (2009) [165] are not expressed (Fig. 3.10). Moreover, GenoGAM consistently had the largest AUC for any gene expression cutoff up to 60% genes to be not expressed (Fig. 3.11). These results indicate that GenoGAM is more sensitive than current methods for testing differential occupancy, while still controlling for type I error rate.

3.3.3.2 Comparison of GenoGAM fit with sliding window smoothing

In the uncommon situation where a benchmark is available as for the Thornton et al. [121] dataset, one can objectively define an optimal window size for sliding window approaches. The log-ratios estimated by GenoGAM fit well to log-ratios computed in sliding windows of size 184bp, the window size maximizing the area under curve for csaw for a gene expression quantile cutoff of 0.15 (Fig. 3.5). Also, the GenoGAM 95% confidence ribbon captures very well the short-range fluctuations of the sliding window estimates. Hence, there is a general agreement between the two approaches. However, the benefits of GenoGAM are clear: First, the GenoGAM fit is smooth and differentiable. Second, unlike in the window-based approach, the amount of smoothing is solely estimated from the ChIP-Seq data, without prior knowledge from the benchmark.

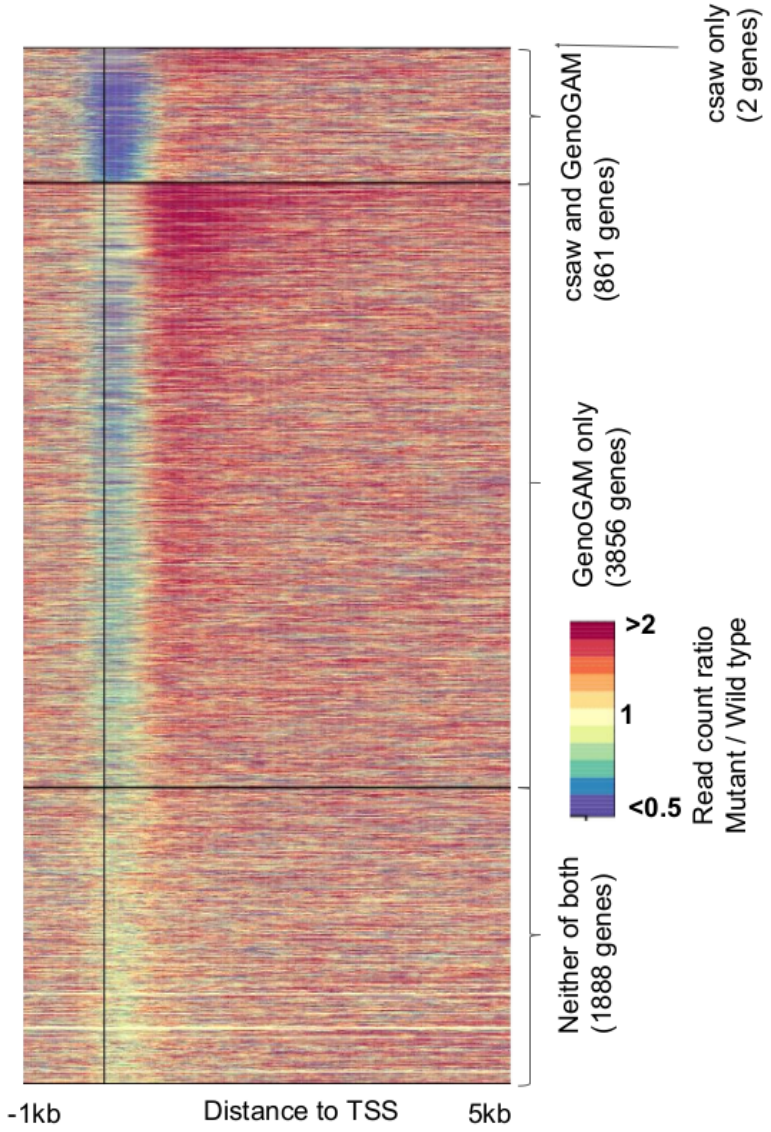

Figure 3.8: Fold change signal across all genes. Fold-change of counts in mutant over wild-type in 150 bp windows for all 6607 yeast genes in the -1 to 5 kb region centered on TSS (vertical black line). The genes are sorted into four groups (separated by the black horizontal lines) according to which method reports them significant. From top to bottom: csaw only (2 genes, not visible), csaw and GenoGAM (861 genes), GenoGAM only (3,856 genes) and none (1,888 genes). Within each group genes are ordered by p-value (lowest to highest from top to bottom). The "csaw and GenoGAM" group is sorted by GenoGAM p-values. Comparisons to all other methods can be found in Appendix. B.1.

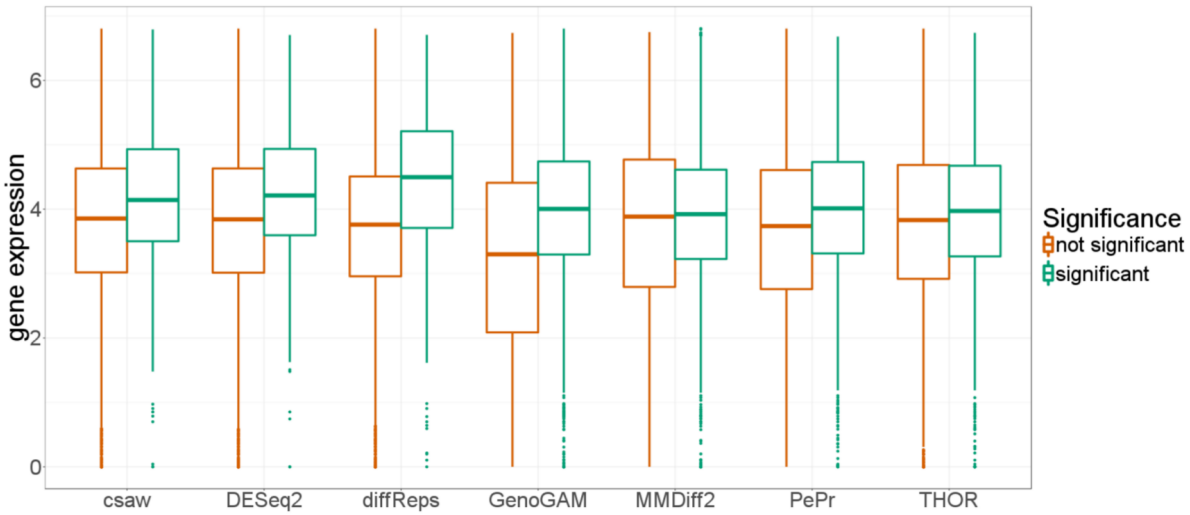

Figure 3.9: Gene expression Boxplots of gene expression levels by method split into the two classes of significant and non-significant genes. Significant genes were called with FDR < 0.1 (GenoGAM, DESeq2, MMDiff2, csaw) or the respective other cutoff specified by the method (THOR, PePr, diffReps). All other genes are regarded as not significant. For each of those two groups a boxplot of gene expression levels is shown. It can be clearly seen (with the exception of MMDiff2) that significant genes (green) seems to be associated with higher gene expression levels.

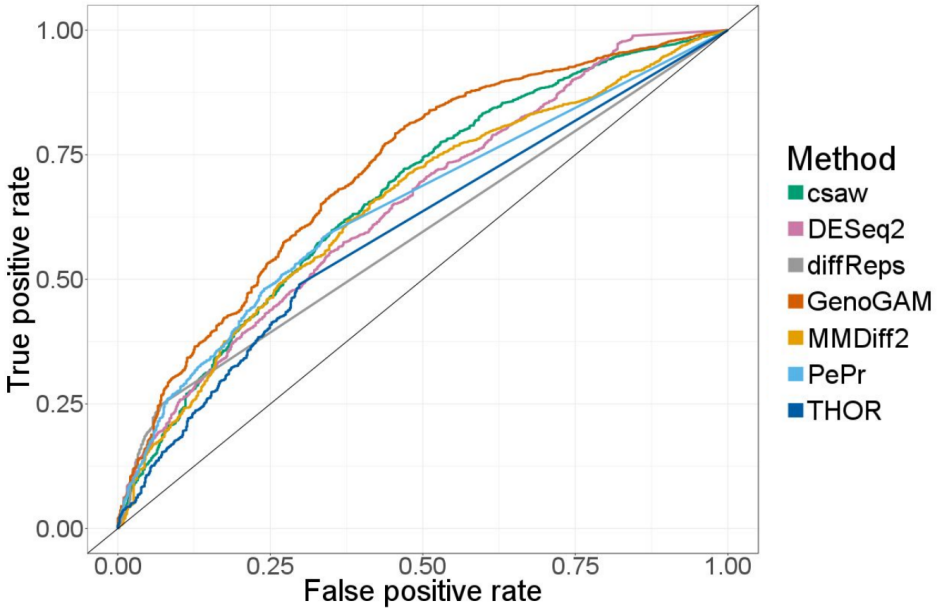

Figure 3.10: ROC curve. ROC curve based on a quantile cutoff of 0.15 (see Appendix B.1). GenoGAM has a constantly higher recall with a lower false positive rate. The partially straight lines for THOR, PePr and diffReps are stemming from tied genes with no significance value.

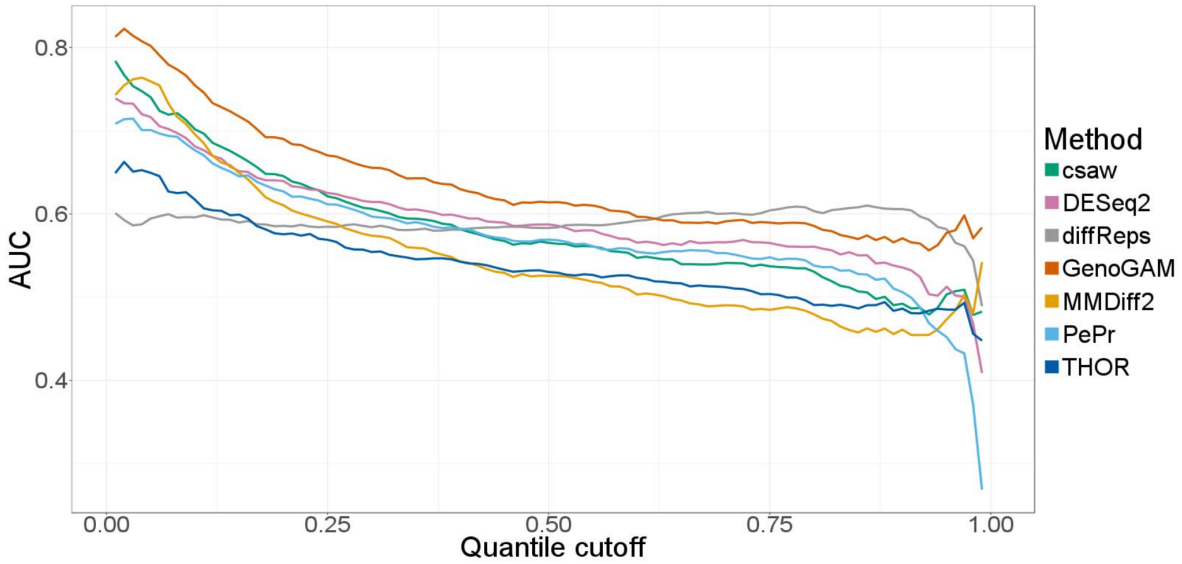

Figure 3.11: Area under the ROC curve. AUC for all possible quantile cutoffs from 0 to 1 in steps of 0.01. Up to a cutoff of 0.6, GenoGAM (red) performs consistently better than all competitor methods by around 0.03-0.04 points above the second best method (csaw and DESeq2, green and pink, respectively). The entire range of quantile cutoffs is shown out of completeness, reasonable values are between 0.15 and 0.25.

4 Scaling GenoGAM

The methodology, results and figures presented in this section are part of the manuscript "GenoGAM 2.0: Scalable and efficient implementation of genome-wide generalized additive models for gigabase-scale genomes" from Stricker et al. 2018 [2].

Section 3.1 introduced the GenoGAM method from a statistical/modeling point of view and illustrated its advantages through a study of differential binding. The results showed increased sensitivity in detecting differential protein occupancies over competing methods, while controlling for type I error rates. However, the primary focus of the first implementation of GenoGAM was a mathematically and statistically sound method that can leverage the underlying generalized additive models framework for genome-wide ChIP-Seq data.

Specifically, it builds on top of the infrastructure provided by the *Bioconductor* software project [166] and the `R` package `mgcv`. The latter is a general-purpose `R` library for fitting GAMs [167] that provides a rich functionality for GAMs with a variety of basis functions, distributions and further features for variable and smoothness selection.

Nonetheless, application of this implementation remains limited in practice to small genomes organisms such as yeast or bacteria, or to selected subsets of larger genomes. A genome-wide fit for the yeast genome (ca. 1 million basis functions, thus 1 million parameters) took 20 hours on a 60-core server. Fits for the human genome could only be restricted to filtered regions or the smallest chromosomes

This section depicts the implementation of a scalable version of GenoGAM, that allows fitting of gigabase-scale genomes. This is achieved by exploiting the sparsity of the model and by using out-of-core data processing. The computing time for parameter and standard error estimation, as well as the memory footprint, is shown to be linear in the number of parameters per tile. The same genome-wide fit for yeast can be obtained in 13 minutes on a standard 8-core desktop machine. Whole-genome fits for human datasets (ca. 300 million parameters) can be obtained in less than 9 hours on the same 60-core server.

4.1 Sparse matrices in GenoGAM

A crucial and possibly expensive step in the IRLS algorithm (Algorithm 1) is the inversion of $(\mathbf{X}^T\mathbf{W}^{[k]}\mathbf{X})$ in order to obtain $\boldsymbol{\beta}^{[k+1]}$. This final matrix product is know as the Hessian matrix $\mathbf{H}$ (or in IRLS the Fisher Information matrix $\boldsymbol{\mathcal{I}}$). In case a penalization term is used the Hessian is computed as:

$$\mathbf{H} = \mathbf{X}^T\mathbf{W}\mathbf{X} - 2\lambda(\mathbf{S} + \epsilon\mathbf{I}) \tag{4.1}$$

If the diagonal matrix $\mathbf{W}$ is computed according to algorithm 1, then plugging equation 4.1 into line seven of algorithm 1 results in the method known as Penalized Iteratively Reweighted Least Squares (P-IRLS) (with $\epsilon\mathbf{I}$ being optional). Because the computation of $\mathbf{H}$ depends on $\mathbf{X}$ and $\mathbf{S}$, its structure depends on them, too.

For each row of the design matrix $\mathbf{X}$ the number of nonzeros is at most 5 times the number of smooth functions because every genomic position x_i is overlapped by 5 cubic B-splines b_r only. Moreover, the penalization matrix $\mathbf{S}$ only has 5 nonzeros per row, as it encodes the second-order difference penalties between coefficients of neighboring splines [155]. Hence, the matrices $\mathbf{X}$ and $\mathbf{S}$, and therefore $\mathbf{H}$, which appears in the majority of the computations via Equations (2.5) and (2.6), are very sparse (see Figure 4.1). This property is extremely useful when it comes to speed up the fitting of the parameters.

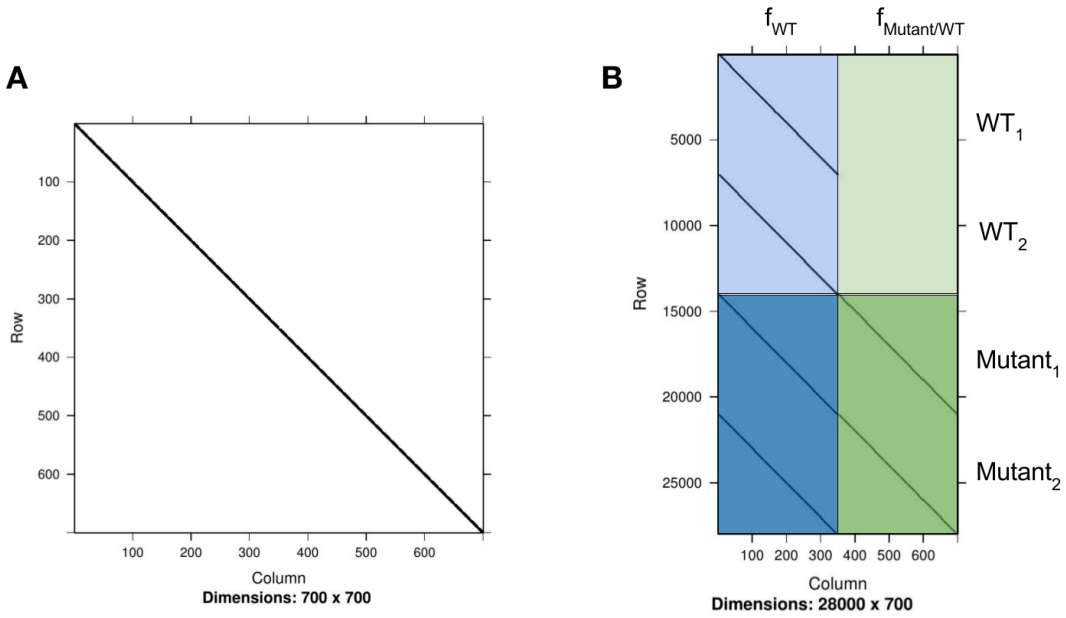

Figure 4.1: Example penalization and design matrix. Depicted are the penalization matrix $\mathbf{S}$ (panel A) and the design matrix $\mathbf{X}$ (panel B) for the differential binding application from section 3.2. It is taken from a region of 7kb, where each smooth function consists of 350 basis functions, i.e. 700 in total. (**A**) The black diagonal band represents values, whereas the white space are zeros. (**B**) The design matrix is block banded with four blocks along the y-axis (the samples) and two blocks along the x-axis (the smooth functions), leading to a total of eight blocks. The black diagonal running lines represent nonzero matrix cells, whereas the rest are zeros. The colors (blue for wildtype smooth and green for mutant over wildtype smooth) and shades (light for wildtype samples and dark for mutant samples) together with the labels help identify and relate the matrix to the experimental design specified in table 3.2.

4.2 Scaling coefficient estimation by Newton-Raphson

Parameters $\boldsymbol{\beta}$ are estimated by maximizing the penalized log-likelihood using the Newton-Raphson iteration (Equation 2.5). Due to the sparsity of the matrices $\mathbf{X}$, $\mathbf{W}$ and $\mathbf{S}$, $\mathbf{H}$ is sparse and cheap to compute. The inverse is never explicitly formed. Instead the linear system is solved by a direct solver using the `SuperLU` library [168]. Furthermore all matrices are stored in a sparse format, avoiding redundant storage of zeros (see section 2.3). The algorithm for a negative binomial distribution with known theta, the log link and penalization term is provided below (Algorithm 2).

The new fitting algorithm differs from the one of `mgcv` in two ways. First, `mgcv` uses QR decomposition of the design matrix $\mathbf{X}$ [151]. However, general QR decomposition destroys the sparse structure of $\mathbf{X}$, making it impossible to exploit it. Investigations in the use of sparse QR decompositions proofed to be less efficient than the final implementation. Second, `mgcv` uses Iteratively Reweighted Least Squares, which employs the Fisher information matrix $\boldsymbol{\mathcal{I}}$, instead of the exact Hessian. The substitute did not lead to any measurable differences in the fitted parameters. It did however improve numerical stability, after abandoning (the more numerically stable) QR decomposition.

Algorithm 2 Newton-Raphson for Negative Binomial

while convergence criteria not met **do**
  $\eta_i^{[t]} \leftarrow \mathbf{X}_i \hat{\boldsymbol{\beta}}^{[t]}$ ▷ Initialize or update values
  $\mu_i^{[t]} \leftarrow \exp(\eta_i^{[t]})$
  $z_i^{[t]} \leftarrow (y_i - \mu_i^{[t]})/(1 + \frac{\mu_i^{[t]}}{\theta})$
  $\mathbf{W}_{ii}^{[t]} \leftarrow \mu_i^{[t]}(1 + \frac{y_i}{\theta})/(1 + \frac{\mu_i^{[t]}}{\theta})^2$ ▷ Compute exact weight matrix $\mathbf{W}^{[t]}$
  $\nabla f(\boldsymbol{\beta}_t) \leftarrow \mathbf{X}^T \boldsymbol{z}^{[t]} - 2\lambda \mathbf{S}\boldsymbol{\beta}^{[t]}$ ▷ Compute first derivative vector
  $\mathbf{H}^{[t]} \leftarrow \mathbf{X}^T \mathbf{W}^{[t]} \mathbf{X} - 2\lambda(\mathbf{S} + \epsilon \mathbf{I})$ ▷ Compute exact Hessian matrix $\mathbf{H}^{[t]}$
  $\hat{\boldsymbol{\beta}}^{[t+1]} \leftarrow \boldsymbol{\beta}_t - (\mathbf{H}^{[t]})^{-1} \nabla f(\boldsymbol{\beta}_t)$ ▷ Solve to obtain $\hat{\boldsymbol{\beta}}^{[t+1]}$
end while

4.3 Scaling standard error estimation by sparse inverse subset algorithm

4.3.1 Structure of the inverse Hessian

The Hessian $\mathbf{H}$ is sparse (see Figure 4.2), but its inverse, the covariance matrix $\mathbf{H}^{-1}$, is usually not. However, the variances of interest (Equation 2.6) can be computed using only a subset of the elements of the inverse $\mathbf{H}^{-1}$. Specifically, denoting for any matrix $\mathbf{A}$:

- $\mathrm{NZ}(\mathbf{A}) = \{(i,j), \mathbf{A}_{i,j} \neq 0\}$ the indices of nonzero elements,

- $C_i(\mathbf{A}) = \{j : \mathbf{A}_{i,j} \neq 0\}$ the column indices of nonzero elements for the i-th row,
- $R_j(\mathbf{A}) = \{i : \mathbf{A}_{i,j} \neq 0\}$ the column indices of nonzero elements for the j-th row,

Theorem 1. *Then* $\boldsymbol{\sigma}^2$ *can be computed only using the elements* $(\mathbf{H}^{-1})_{l,j}$*, where* $(l, j) \in \mathrm{NZ}(\mathbf{H})$

Proof. On the one hand, we have:

$$\begin{aligned} \sigma_i^2 &= \sum_{l,j} \mathbf{X}_{i,l}(\mathbf{H}^{-1})_{l,j}\mathbf{X}_{i,j} \\ &= \sum_{(l,j)\in C_i^2(\mathbf{X})} \mathbf{X}_{i,l}(\mathbf{H}^{-1})_{l,j}\mathbf{X}_{i,j} \end{aligned} \tag{4.2}$$

On the other hand, Equation 4.1 implies that $\mathrm{NZ}(\mathbf{H}) = \mathrm{NZ}(\mathbf{X}^T\mathbf{W}\mathbf{X}) \cup \mathrm{NZ}(\mathbf{S}) \cup \mathrm{NZ}(\mathbf{I})$. Since

$$(\mathbf{X}^T\mathbf{W}\mathbf{X})_{l,j} = (\sum_i \mathbf{X}_{i,l}\mathbf{W}_{i,i}\mathbf{X}_{i,j}), \tag{4.3}$$

it follows that:

$$(\mathbf{X}^T\mathbf{W}\mathbf{X})_{l,j} \neq 0 \Leftrightarrow \exists i, i \in R_l(\mathbf{X}) \text{ and } i \in R_j(\mathbf{X}) \Leftrightarrow \exists i, (l,j) \in C_i^2(\mathbf{X})$$

Moreover, the nonzeros of the identity matrix $\mathbf{I}$ is a subset of the nonzeros of the second-order differences penalization matrix $\mathbf{S}$ [155]. Furthermore, the nonzeros of the second-order differences penalization matrix $\mathbf{S}$, which penalizes differences between triplets of consecutive splines, is a subset of the nonzeros of $\mathbf{X}^T\mathbf{X}$, since genomic positions overlap five consecutive splines when using cubic B-splines. Hence, $\mathrm{NZ}(\mathbf{H}) = \{(l, j), \exists i, (l, j) \in C_i^2(\mathbf{X})\}$. Together with Equation 4.2, this proves the result.

□

Using only the elements of $\mathbf{H}^{-1}$ that are in $\mathrm{NZ}(\mathbf{H})$ applies to computing the variance of any linear combinations of the $\boldsymbol{\beta}$ based on the same sparse structure of $\mathbf{X}$ or a subset of it. Hence, it applies to computing the variance of the predicted value for any smooth function $f_k(x)$ or computing the variance of the derivatives of any order r of any smooth $\frac{d^r f_k(x)}{d^r x}$. The former is particularly important, because computation of standard errors according to equation 2.6 or 4.1 will yield standard errors of the full response fit. However, one is usually more interested in the fold-change represented by one or multiple smooths, e.g. $f_{mutant/WT}$ in the differential binding application. Hence only a subset of $\mathbf{H}$ and $\mathbf{X}$ is required for the computation. For instance, in the differential binding application only the right side of $\mathbf{X}$ (the complete green part in figure 4.1B) and the lower right corner block of $\mathbf{H}$ (the green part in figure 4.2) is needed to compute the variance of the smooth function $f_{mutant/WT}$.

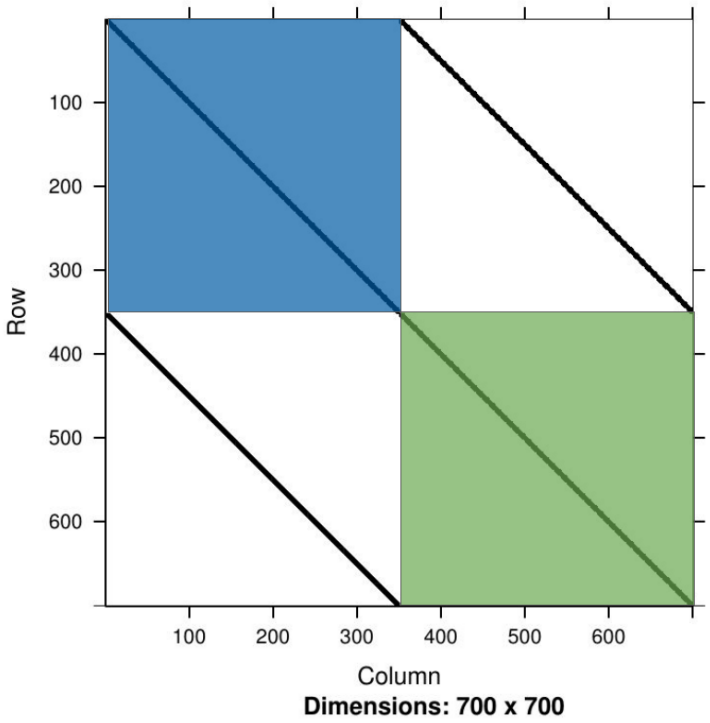

Figure 4.2: Example Hessian matrix. Depicted is the Hessian matrix $\mathbf{H}$ for the differential binding application from section 3.2. It is taken from a region of 7kb, where each smooth function consists of 350 basis functions, i.e. 700 in total. The Hessian matrix is block banded with four blocks in total. Where each block represents the inverse covariance matrix (also known as the precision matrix) between the smooth functions. The black diagonal running lines represent nonzero matrix cells, whereas the rest are zeros. The colors (blue for background, green for Mutant/WT) help identify the inverse variance (that is the covariance of a smooth with itself).

4.3.2 Exact standard error computation by the sparse inverse subset algorithm

To obtain the elements of $\mathbf{H}^{-1}$ that are in NZ($\mathbf{H}$), the *sparse inverse subset algorithm* in combination with the sparse Cholesky decomposition was used [169]. Given a sparse Cholesky decomposition of symmetric matrix $\mathbf{A} = \mathbf{L}\mathbf{L}^T$, the sparse inverse subset algorithm returns the values of the inverse $\mathbf{A}^{-1}$ that are nonzero in the Cholesky factor $\mathbf{L}$. Since the set of nonzero coordinates in the lower triangle of $\mathbf{A}$ is a subset of the nonzero coordinates in the Cholesky factor $\mathbf{L}$ [156], the sparse inverse subset algorithm provides the required elements of $\mathbf{H}^{-1}$ when applied to a sparse Cholesky decomposition of $\mathbf{H}$.

The sparse inverse subset algorithm is concerned with computing the sparse inverse subset, called *Zsparse*, of a symmetric matrix $\mathbf{A}$ through LU or Cholesky factorization and the use of so called *Takahashi equation(s)* to relate those factors to $\mathbf{Z} = \mathbf{A}^{-1}$ [169].

Zsparse is defined as the set of entries in the upper part of $\mathbf{A}^{-1}$ that are given by the location of non-zero entries in the factorized matrix:

$$\text{Zsparse} = \{Z_{ij} | (\mathbf{U})_{ij} \neq 0\} \subseteq \mathbf{Z} \tag{4.4}$$

In case of Cholesky decomposition it can also be stated as:

$$\text{Zsparse} = \{Z_{ij} | (\mathbf{L})_{ij} \neq 0\} \subseteq \mathbf{Z} \tag{4.5}$$

Then the Takahashi equation to compute $\mathbf{Z}$ can be stated as follows [169]:

$$Z_{ij} = \frac{\delta_{ij}}{L_{ii}} - \sum_{k \in \mathcal{M}(i)}^{n} L_{ki} Z_{kj}, \quad j \geq i,\, i = n, \ldots, 1 \tag{4.6}$$

with $\mathcal{M}(i)$ the set of those k where L_{ki} is nonzero:

$$\mathcal{M}(i) = \{k > i : L_{ki} \neq 0\} \tag{4.7}$$

and $\delta_{ij} = 1$ if $i = j$ and zero otherwise. Note, that Z_{ij}/Z_{kj} can be found on both sides of the equation. Therefore its values have to be computed recursively starting with Z_{nn}. Furthermore, equation 4.6 actually only computes the lower triangle, which is sufficient due to symmetry.

The standard error computation is gaining computational speedup from two algorithms (sparse Cholesky decomposition and sparse inverse subset algorithm) which rely on the same principles of the symmetric multifrontal method [170]. This method is based on the use of dense submatrices called *frontal matrices* that are formed as the multifrontal algorithm progresses.

In general, the symmetric multifrontal algorithm consists of a symbolic analysis phase and a numerical factorization phase [156]. In the analysis phase a fill-reducing pivot ordering algorithm is used to establish the pivot order and data structures. In addition, the relationships among the frontal matrices are established and given by the *assembly tree* [171]. The numerical work to actually compute the Cholesky factors is done in the numerical factorization phase. The assembly tree is used to guide the computation in this phase [169]. That is, the order in which the dense submatrices are factorized. The sparse inverse subset algorithm works in a similar way but with inverted objects. Here, *inverted* refers to the fact, that tree construction and computation is not executed from 1 to n, but from n to 1 (see Equation 4.6). Thus, first the *inverse assembly tree* is constructed, which establishes the connections between the *inverse frontal matrices*. Then, those dense submatrices are inverted guided by the inverse assembly tree, evaluating a matrix-vector form of the Takahashi equations [169]. For a detailed overview over the method and other sparse matrix methods see Campbell and Davis (1995) [169] and Davis (2006) [156]. See also Rue [172] for similar ideas for Gaussian Markov Fields.

On the implementation side the sparse inverse subset algorithm is performed, using the `R` package `sparseinv` [173], itself a wrapper of relevant code from the `SuiteSparse` software [174]. Once the sparse inverse subset of the Hessian is obtained, σ_i^2 can be computed according to Equation (2.6) with a slight improvement: Because only the

diagonal from the final matrix product is needed, the implementation does not perform two matrix multiplications. Instead, only the first product is computed, then multiplied element-wise with $\mathbf{X}_k^T$ and summed over the columns.

4.4 Complete GenoGAM 2.0 Workflow

Additionally to the algorithmic improvements, the implementation of the second version of GenoGAM (GenoGAM 2.0) involves an optimized Hierarchical Data Format (HDF5) backend for data storage [175]. The complete workflow can then be described as follows:

Data preprocessing consists of reading raw read alignments from BAM files, centering the fragments, computing the coverage vector $\boldsymbol{y}$, and splitting the data by genomic tiles (Figure 4.3). Afterwards, normalization factors for sequencing depth variation are computed using DESeq2 [141]. Preprocessed data is stored in HDF5 files through the `R` packages `HDF5Array` [176] and `rhdf5` [177]. This allows writing in parallel as the data is being preprocessed, which reduces the memory footprint of this step. For all subsequent matrix operations the `Matrix` [178] package is used, which implements routines for storage, manipulation and operations on sparse matrices.

Fitting GenoGAM models on tiles is achieved by the Newton-Raphson algorithm (Algorithm 2 and section 4.2). This is done on few representative tiles during cross-validation in order to identify optimal smoothing and dispersion parameters λ and θ, and subsequently when fitting the model on the full dataset.

The variance of the smooth estimates (Equation 2.6) is obtained using the sparse inverse subset algorithm as detailed in subsection 4.3.2. The implementation is based on the `R` package `sparseinv` [173], which wraps relevant code from the `SuiteSparse` software [174]. As in the previous GenoGAM model [1], fitting on different tiles is conducted in parallel. The result objects for the fits, variances and parameters are initialized prior to fitting on hard drive. This allows the processes to write results in parallel on the fly, ensuring fast computation and low memory footprint. The HDF5 storage is further optimized for reading time by adjusting HDF5 chunk size to the size of the tiles (for preprocessed count data) and chunks (for fits and variance). As HDF5 is not process-safe on `R` level, writing is serialized by a queuing mechanism.

The parallelization backend is provided by the `R` package `BiocParallel` [179]. It offers an interface to a variety of backends and can be registered independently of GenoGAM. Parallelization is performed over chromosomes during the read-in process. Over tuples of folds and tiles during cross-validation process and over tiles during fitting process. Because some backends have a particular long start-up time, the use of many processes might end up dominating computation time. Specifically during cross-validation on small and limited number of regions, this might pose a problem. Therefore an optimal number of workers is automatically obtained and registered by the cross-validation function and reset on exit.

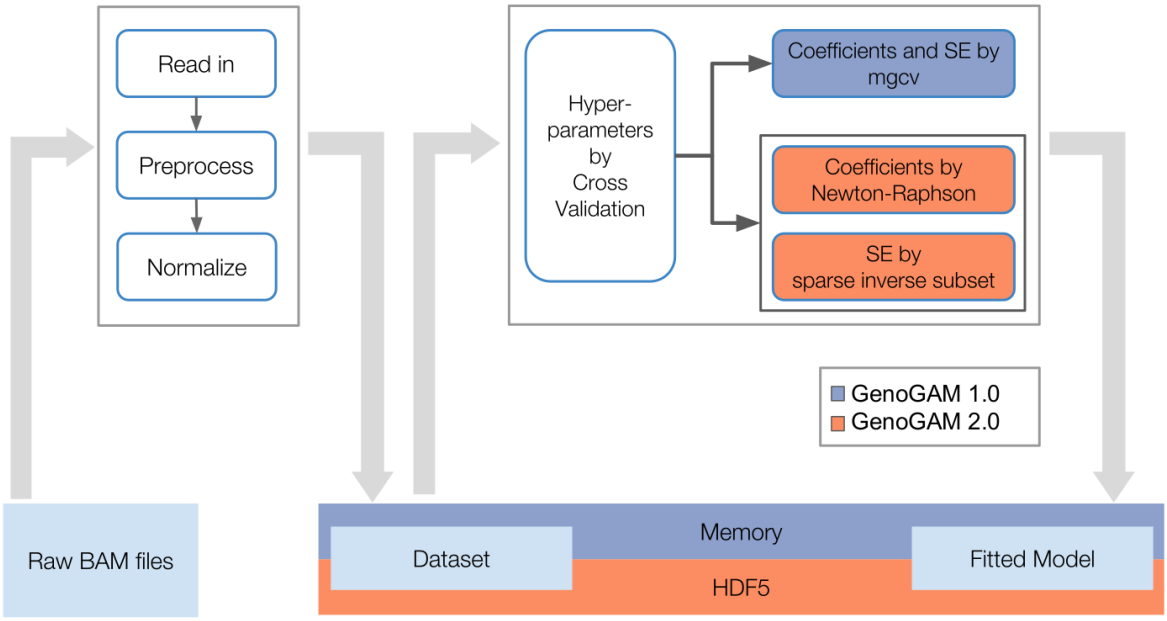

Figure 4.3: Schematic overview highlighting the difference between GenoGAM 1.0 and GenoGAM 2.0: Raw BAM Files are read-in, pre-processed, normalized and written to hard drive in HDF5 format. Moreover, normalization factors for sequencing depth variation are computed using DESeq2 [141]. The resulting object is the dataset upon which fitting is done. Then, global hyperparameters λ and θ are estimated by cross-validation and for each tile coefficients are estimated via Newton-Raphson and standard errors via sparse inverse subset algorithm. The final model is written as a new object to hard drive in HDF5 format. Note, that the schematic view is a simplification: The pre-processed dataset and the fitted model are not generated in memory and written to HDF5 in the end. Instead, all HDF5 matrices are initialized on hard drive directly and the writing is done on the fly. Blue (GenoGAM 1.0) and orange colors (GenoGAM 2.0) mark differences between both GenoGAM versions.

4.5 Runtime and memory footprint results

4.5.1 Results on coefficient estimation

Figure 4.4 displays the comparison in fitting runtime (A) and memory usage (B) of the Newton-Raphson method (section 4.2) versus the method underlying the previous GenoGAM version on a single core. Computation was capped at approximately 2 hours, which leads the blue line (GenoGAM 1.0) to end after around 1,100 parameters. It can be clearly seen, that exploiting the advantages of the data structures leads to improvements by 2 to 3 orders of magnitude. At the last comparable point at 1,104 parameters it took the previous method 1 hours and 37 minutes, while it was only 1 second for the Newton-Raphson method. This number increased a little bit towards the end to almost 5 seconds for 5,000 parameters.

Additionally, the more efficient storage of sparse matrices and the lightweight implementation reduces the overhead and memory footprint. Again at the last comparable point, the memory used by the previous method is 8 Gbyte while it is 52 MByte by the new method, increasing to 250 MByte at the 5,000 parameters mark. Moreover, runtime per tile drops empirically from growing cubically with the number of parameters in GenoGAM 1.0 to linearly in GenoGAM 2.0. The memory footprint drops empirically from growing quadratically with the number of parameters in GenoGAM 1.0 to linearly in GenoGAM 2.0 (dashed black lines fitted to the performance data).

4.5.2 Results on standard error estimation

Alternatively to the direct computation of the inverse Hessian with consecutive computation of variance vector $\boldsymbol{\sigma}^2$ (Equation 2.6), it is also possible to directly compute $\boldsymbol{\sigma}^2$. Here and hereafter the smooth function specific index k is dropped for simplicity. In a comment to the paper of Lee and Wand [180], a direct way to compute $\boldsymbol{\sigma}^2$ without inverting $\mathbf{H}$ was proposed by Simon Wood ([181]). The comment states, that in general, if $y = \mathbf{X}\boldsymbol{\beta}$, then

$$\sigma_i^2 = \sum_{j=1}^{p}((\mathbf{X}\mathbf{P}^T\mathbf{L}^{-1})_{i,j})^2 \tag{4.8}$$

Where $\mathbf{P}$ is the permutation matrix and $\mathbf{L}^{-1}$ is the inverted lower triangular matrix resulting from Cholesky decomposition of $\mathbf{X}^T\mathbf{H}^{-1}\mathbf{X}$.

Figure 4.5 shows the comparison of both methods in time and memory on a single core, with the above proposed method depicted as "indirect" (blue). While both methods have linear memory footprint, the slope of the indirect method is around four times higher. The computation time is significantly in favor of the sparse inverse algorithm. This is because for every σ_i^2 a triangular system has to be solved to obtain $(\mathbf{X}\mathbf{P}^T)_i\mathbf{L}^{-1}$. Although solving the complete system at once is faster, it had a high memory consumption when it came to increased number of parameters in the test implementation. Thus the performance presented is based on batches of σ_i^2 to obtain a fair trade-off

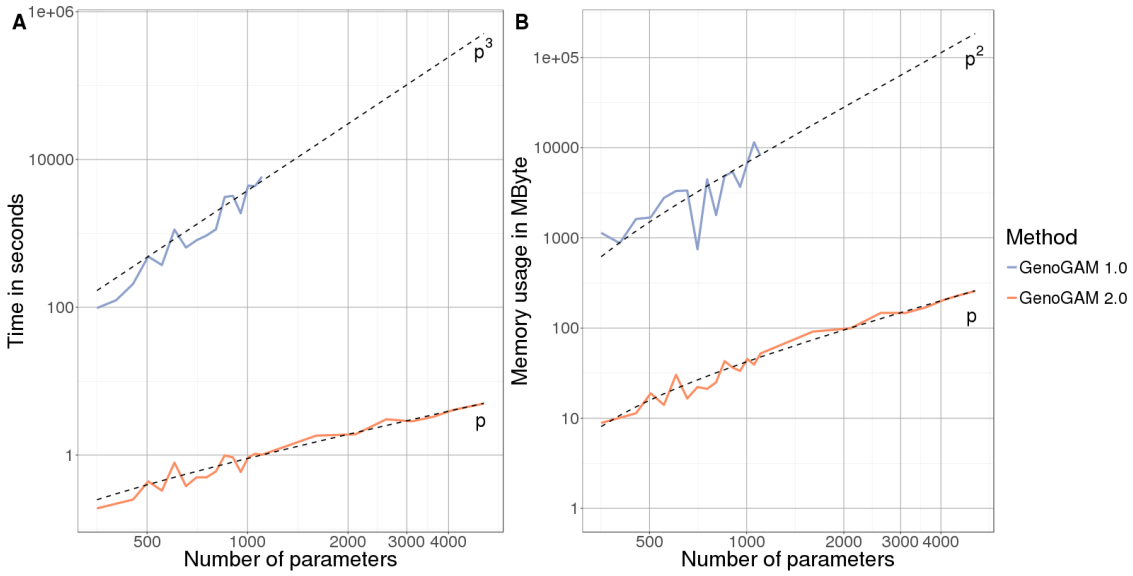

Figure 4.4: Parameter estimation performance. (**A**) Empirical runtime for the estimation of parameter vector $\boldsymbol{\beta}$ is plotted in log-scale against increasing number of parameters (also log-scale). The runtime is capped at around 2 hours, such that runtime of GenoGAM 1.0 (blue line) terminates after 1,100 parameters. GenoGAM 2.0 (red line) achieves linear runtime in p (dotted line p), the number of parameters, compared to the previous cubic complexity (dotted line p^3). (**B**) Memory consumption in MByte for the estimation of coefficients vector $\boldsymbol{\beta}$ is plotted against number of parameters (also log-scale). Due to the runtime cap at around 2 hours the runtime of previous GenoGAM version (blue line) does terminate after 1,100 parameters. The storage of matrices in sparse format and direct solvers avoiding full inversion keep the memory footprint low and linear in p (dotted line p) in GenoGAM 2.0 (red line) compared to quadratic in GenoGAM 1.0 (blue line, dotted line p^2).

between runtime and memory. Nevertheless, the difference remains around 2 orders of magnitude. Moreover runtime goes now linearly in practice for the sparse inverse subset algorithm compared to quadratically for the indirect method (dashed black lines fitted to the performance data).

4.5.3 Performance on human and yeast ChIP-Seq datasets

The previous version of GenoGAM could only be partially applied genome-wide for megabase-scale genomes such as the yeast genome and was impractical for gigabase-scale genomes such as the human genome. A genome-wide model fit with two conditions and two replicates each took approximately 20 hours on 60 cores [1]. With computational and numerical improvement on the one side and a data model largely stored on hard drive on the other side, runtime and memory requirements have dropped significantly. Figure 4.6 shows the runtime performance on seven human ChIP-Seq datasets with two replicates for the IP and one or two replicates for the control. The analysis was

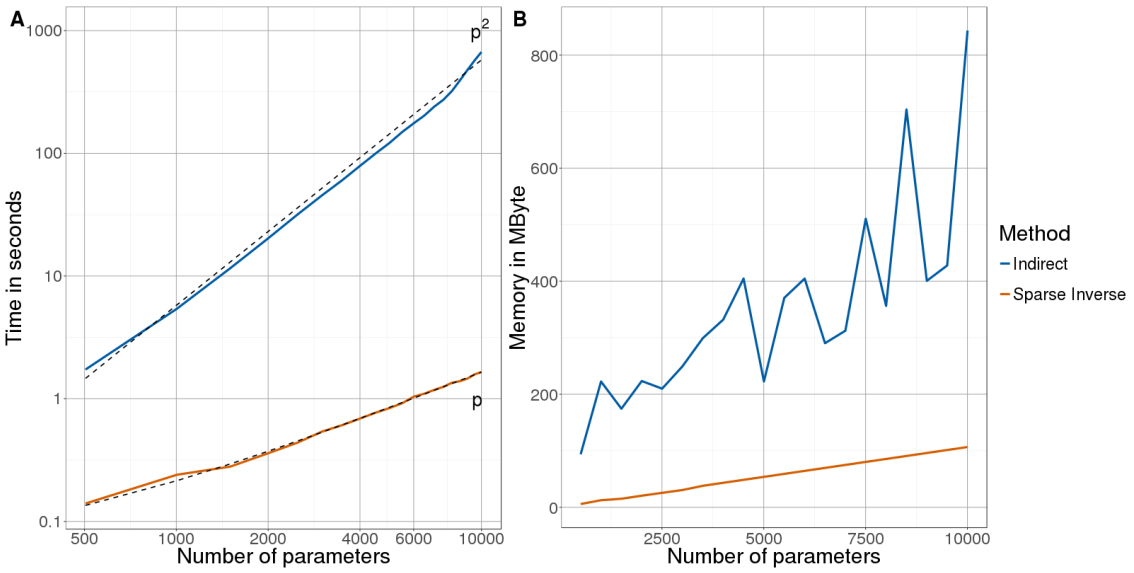

Figure 4.5: Standard error computation. (**A**) Empirical runtime for the computation of standard error vector $\boldsymbol{\sigma}^2$ is plotted in log-scale against increasing number of parameters (also log-scale). Computation based on sparse inverse subset algorithm (red line) achieves linear runtime in p (dotted line p), the number of parameters, compared to quadratic complexity (dotted line p^2) of the "indirect" method (blue line). (**B**) Memory consumption in MByte for the computation of standard error vector $\boldsymbol{\sigma}^2$ is plotted against number of parameters. Though both methods achieve linear memory consumption in p, the slope of the "indirect" method (blue line) is around 4 times greater than of the sparse inverse subset algorithm (red line). A consequence likely due to the recursive computation of the inverse instead of solving of a triangular system.

performed with 60 cores on a cluster, the memory usage never exceeded 1.5GB per core and was mostly significantly lower. The overall results show that around 20 minutes are spent with pre-processing the data, which is largely occupied by writing the data to HDF5 files. One hour of cross-validation, to find the right hyperparameters and around 7 to 8 hours of fitting, amounting to a total runtime of 8 to 9 hours. It is also notable, that the transcription factors NRF1, MNT and FOXA1 include two controls instead of one, thus efficiently increasing the amount of data to fit by a third, but the runtime by around 40 minutes.

Additionally, the same yeast analysis is shown by running on a laptop with 8 cores for comparison to the previous version. The total runtime is around 13 minutes with the cross-validation significantly dominating both other steps (around 9 minutes). This is due to the fact that the number of regions used is fixed at 20, resulting in 200 model fitting runs for one 10-fold cross-validation iteration. Hence, for a small genome like the yeast genome, hyperparameter optimization may take more time than the actual model fitting. Note, that during cross-validation the only difference between human and yeast analysis is the underlying data and the parallel backend. However the runtime on yeast

is only 1/6 of the runtime in human. Both factors play a role in this: First, the parallel backend in the yeast run uses the `Multicore` backend, allowing for shared memory on one machine. While the human run uses the `Snow` (**s**imple **n**etwork **o**f **w**orkstations) backend, which needs to initiate the workers and copy the needed data first, resulting in an overall greater overhead. Second, convergence on yeast data is generally faster due to higher coverage resulting not only in less iterations by the Newton-Raphson, but also during cross-validation.

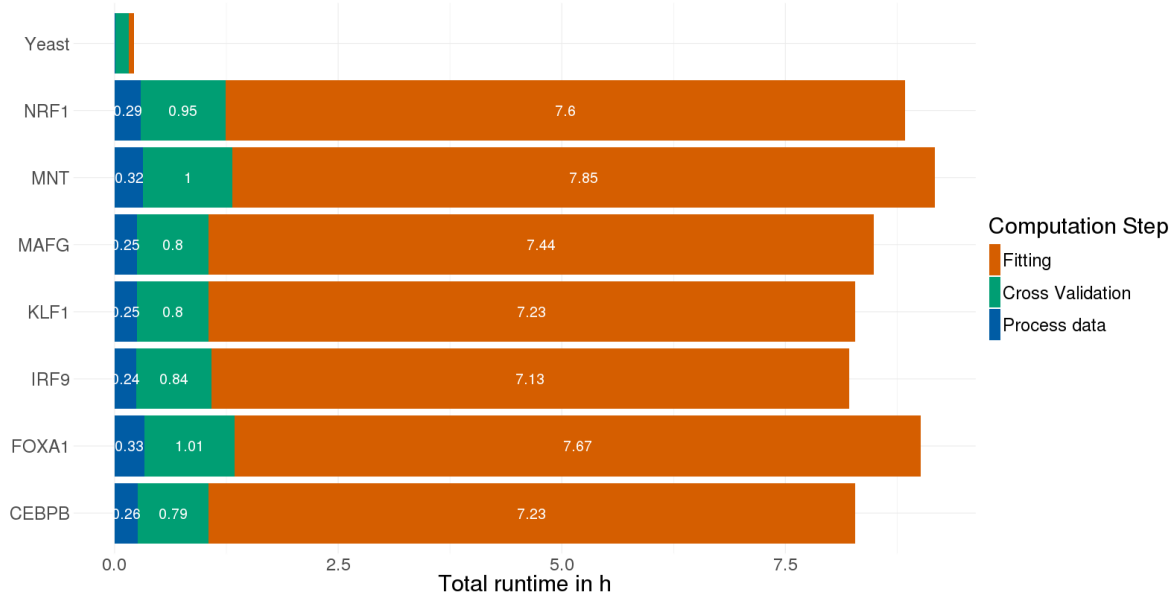

Figure 4.6: Genome-wide performance for human and yeast. The performance of GenoGAM 2.0 on seven human ChIP-Seq datasets for the transcription factors NRF1, MNT, FOXA1, MAFG, KLF1, IRF9 and CEBPB. The first three of which contain two replicates for the control, while the rest contains only one. This increases the data by around a 1/3, but the runtime by around 40 minutes, equivalent to approximately 1/11. Overall ca. 20 minutes are spent on data processing (blue), up to one hour on cross-validation (green) and 7 - 8 hours of fitting (red) amounting to a total of 8 - 9 hours runtime on 60 cores, with the `snow` parallel backend and HDF5 data structure. At the very top yeast runtime is shown on a regular machine with 8 cores, the `Multicore` backend and all data kept in memory avoiding I/O to hard drive. Data processing (blue, almost not visible) takes 40 seconds, cross-validation around 9 minutes (green) and fitting 3.5 minutes (red).

4.5.4 Replication of previous benchmark analyses show equivalent biological accuracy

To demonstrate that GenoGAM 2.0 leads to the same results than GenoGAM 1.0 we have re-generated benchmark analyses of the first paper [1]. The first benchmark is a

differential occupancy application that demonstrates that GenoGAM has greater sensitivity for same specificities than alternative methods. Figure 4.7 is a replication of figure 3.10 and figure 4.8 the replication of figure 3.11 with GenoGAM 2.0 added.

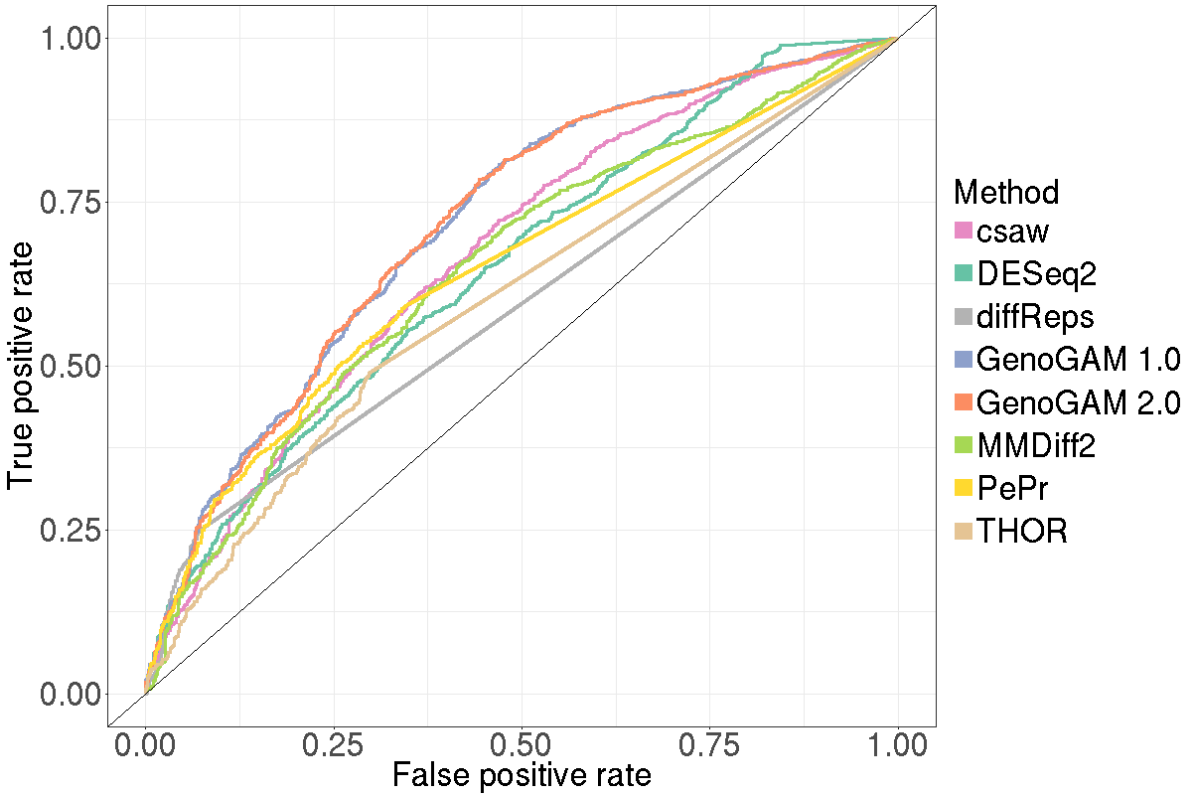

Figure 4.7: Replication of figure 3.10 with GenoGAM 2.0. ROC curve based on a quantile cutoff of 0.15 (15% of the genes are assumed to be true negatives). GenoGAM (orange and blue) has a constantly higher recall with a lower false positive rate.

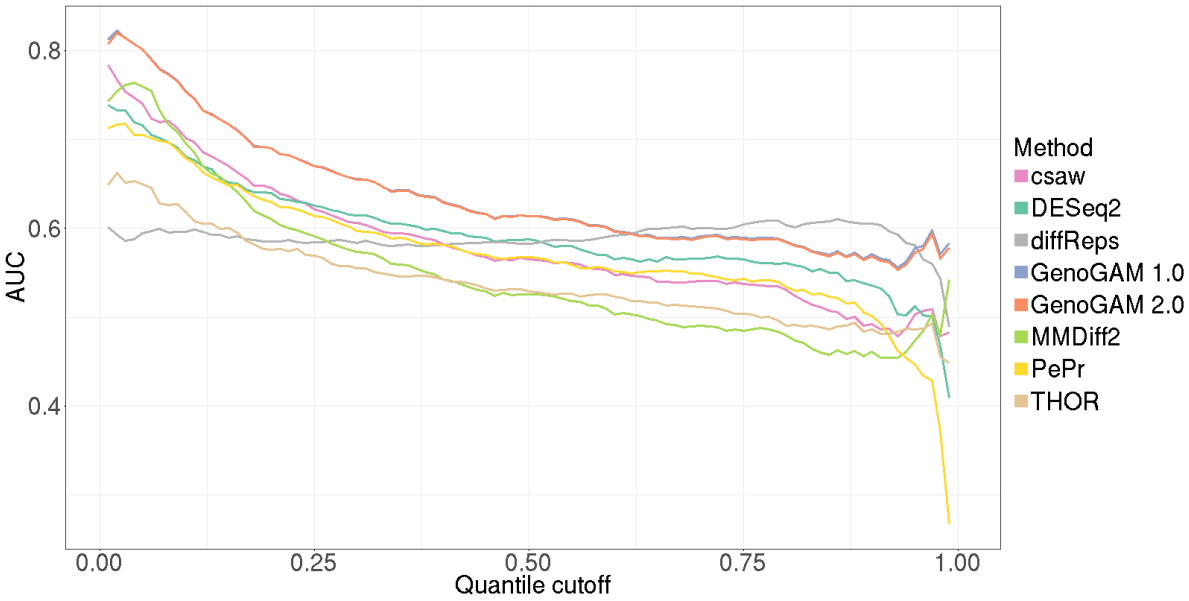

Figure 4.8: Replication of figure 3.11 with GenoGAM 2.0. Area under the curve (AUC) for all possible quantile cutoffs from 0 to 1 in steps of 0.01. GenoGAM 1.0 (blue) and GenoGAM 2.0 (orange) are almost identical and are thus largely overlapping. Up to a cutoff of 0.6, GenoGAM (orange and blue) performs consistently better than all competitor methods by around 0.03-0.04 points above the second best method (csaw and DESeq2, pink and green, respectively). The entire range of quantile cutoffs is shown out of completeness, reasonable values are between 0.15 and 0.25.

5 From differential occupancy to other applications

The methodology, results and figures presented in this section are part of the manuscript "GenoGAM: genome-wide generalized additive models for ChIP-Seq analysis" from Stricker and Engelhardt et al. 2017 [1].

5.1 Peak calling

5.1.1 Building a peak caller

The smooth function estimates and their representation as P-splines provided by GAM offer new opportunities for subsequent analyses apart from differential binding: First and second order derivatives can be computed immediately (Fig. 5.1a). Those can be used to infer summits of ChIP-Seq peaks (as positions x where $f'(x) = 0$ and $f''(x) < 0$). Figure 5.1 illustrates the construction of such a peak caller. To assess statistical significance of the peak heights, an empirical z-score is introduced that contrasts the estimate of the log-occupancy μ at the peak to a robust estimate of background log-occupancy level μ_0, taking both background level variance σ_0^2 and uncertainty of peak height σ^2 into account:

$$z = \frac{\mu - \mu_0}{\sqrt{\sigma^2 + \sigma_0^2}}. \tag{5.1}$$

In order to account only for the background without potential peaks, μ_0 is estimated as the *shorth* from the Bioconductor `genefilter` package for all $f(x_i)$, $i = 1, \ldots, n$ (midpoint of the shortest interval containing half of the data) of all fitted values. The fitted values smaller than the shorth are mirrored on it, such that a symmetric density is created that excludes the values larger than the shorth, in particular those high values in the right side tail representing peaks (Fig. 5.1b). The variance of this newly created distribution is then estimated in a robust fashion by the median absolute deviation (MAD) giving σ_0^2.

Borders can be obtained by using the pointwise z-score around peak summits with the respective cutoff. However, this is not advisable, since peak width depends on ChIP-Seq fragment size. A fixed distance around the summit (e.gg +/-200 bp) can be as justifiable. Regarding significance, a practical approach to model the null distribution of peak scores is to assume that false positive peaks arise from symmetric fluctuations of the background and thus distribute similarly to local minima, or peaks found when

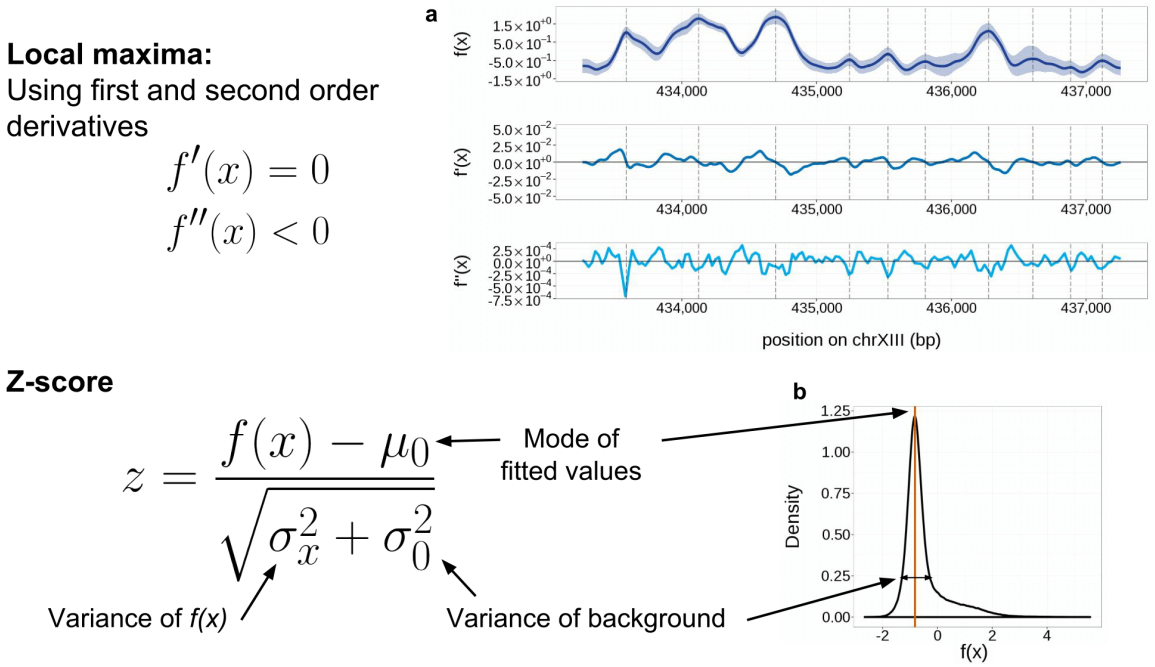

Figure 5.1: Construction of a peak caller. (**a**) Fit of protein log-occupancy $f_{\text{protein}}(x)$ (top), its first derivative $f'_{\text{protein}}(x)$ (middle) and its second derivative $f''_{\text{protein}}(x)$ (bottom). Same data as in Figure 3.1 is used (the left half), but extended a little further downstream. (**b**) Illustration of the z-score computing procedure. See detailed description in paragraph above.

inverting the role of input and IP [104]. Therefore false discovery rate is estimated using the z-score distributions of the local minima. That is, peaks are called on $-f_{\text{protein}}$ (so called *valleys*). Their z-scores are obtained by recomputing μ_0 and σ_0 and applying the same formula. The FDR for a given minimum z-score z is estimated by $\frac{|V_z|}{|P_z|}$, where P_z and V_z are the sets of peaks and valleys, respectively, with a z-score greater than or equal to z.

Comparison of this approach to a few widely used peak callers (MACS [104], JAMM [145] and ZINBA [97]) was performed on small size datasets (Human chromosome 22 and yeast).

5.1.2 The model

The TFIIB yeast dataset consisted of two samples: one input and one IP without replicates. Therefore the following GenoGAM model was used for TFIIB yeast data:

$$y_i \sim \text{NB}(\mu_i, \theta) \tag{5.2}$$

$$\log(\mu_i) = f_{\text{input}}(x_i) + f_{\text{protein}}(x_i) z_{j_i,\text{protein}}, \tag{5.3}$$

where $z_{j_i,\text{protein}} = 1$ whenever j_i is the index of an IP sample and $z_{j_i,\text{protein}} = 0$ whenever j_i is the index of an input sample. Since no replicates were present, there was no need for an offset.

Human datasets contained two biological replicates for the protein samples and at least one input sample. However, the library sizes of the input samples were too low for a robust analysis. Therefore the following GenoGAM model was used for the human datasets:

$$y_i \sim \text{NB}(\mu_i, \theta) \tag{5.4}$$

$$\log(\mu_i) = \log(s_{j_i}) + f_{\text{protein}}(x_i), \tag{5.5}$$

where the offsets $\log(s_{j_i})$ are log-size factors computed to control for sequencing depth variation between replicates. Contrary to the differential binding application where all tiles were used in offset computation, in peak calling only top 1,000 tiles with smallest p-value are selected. The p-values are determined by DESeq2 test for enrichment of IP over input performed on total read counts per tile. This allows to select tiles that were most likely containing peaks. Since in this case input was dropped from the model, determination of p-values was not possible, tiles with the highest sum were used.

5.1.3 Benchmark and results

Benchmarking was conducted based on the correct identification of motifs by the peak caller. In order to compute distances from peaks to motifs, peak summits and motif centers were used. Only peaks with summits within 500 bp of a motif border (left or

right) were considered. Otherwise the peak was regarded as due to other biological process or fluctuation. Of these peaks, those within 30bp of the motif center were regarded as correctly identified. For detailed competitor method specification refer to appendix A.3.3.

5.1.3.1 Result on yeast dataset

First, the performance of GenoGAM, MACS [104], JAMM [145] and ZINBA [97] was compared in identifying binding sites of TFIIB in an in-house dataset of S.cerevisiae version 2 (sacCer2, see Appendix A.3.1 for experimental protocol). For about 20% of yeast promoters, recruitment of TFIIB is triggered by the well-characterized DNA element TATA-box, providing at these promoters a ground truth for a TFIIB occupancy peak [182]. In total 1,105 TATA-boxes were mapped genome-wide by regular expression of a consensus motif and considered 1 kb regions centered on TATA-boxes for benchmarking (see Appendix A.3.2 for identification of TATA-box locations).

In these regions, significant peak summits (FDR < 0.1) from GenoGAM were substantially closer to TATA-boxes than those of alternative methods (median absolute distance 58 bp, third quartile 144 bp for GenoGAM versus 152 and 247 bp for MACS, 82 and 174 bp for JAMM, and 155 and 237 bp for ZINBA, respectively Fig. 5.2a). Moreover, the proportion of peak summits within 30 bp of a TATA-box center was twice as high as for any other method independently of the number of reported peaks (Fig. 5.2b), showing that the improvement was robust to the score threshold.

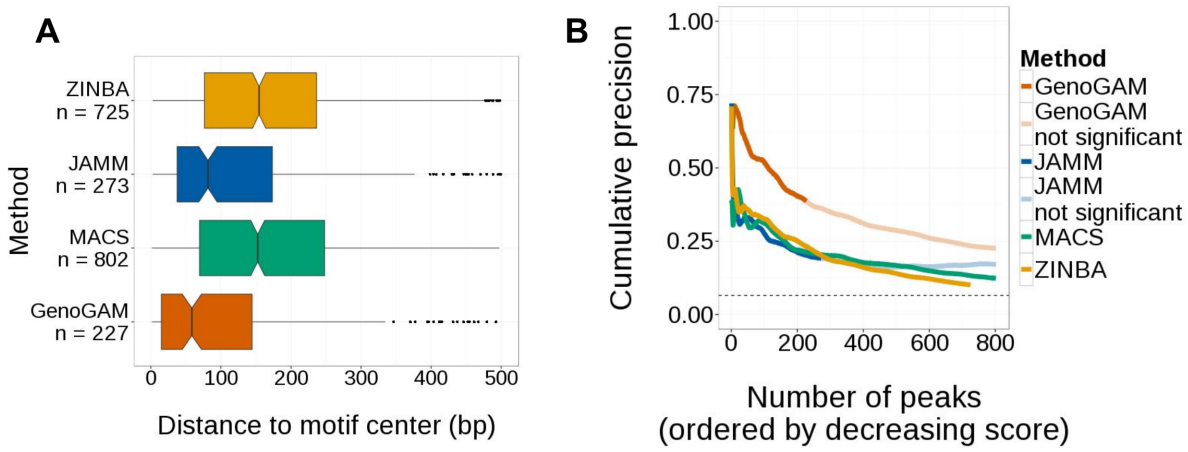

Figure 5.2: GenoGAM peak caller comparison with state-of-the-art methods (**A**) Boxplot of distances between significant peaks (FDR < 0.1) and TATA box for the yeast TFIIB dataset (Appendix A.3.1) for GenoGAM (orange), MACS (green) and JAMM (blue) and ZINBA (yellow). (**B**) Proportion of TFIIB peaks (y-axis) within 30 bp of a TATA box for GenoGAM (orange), MACS (green), JAMM (blue) and ZINBA (yellow) versus number of selected peaks when ordered by decreasing score (x-axis). For each method transparent colors indicate peaks that the method considers not significant (FDR > 0.1).

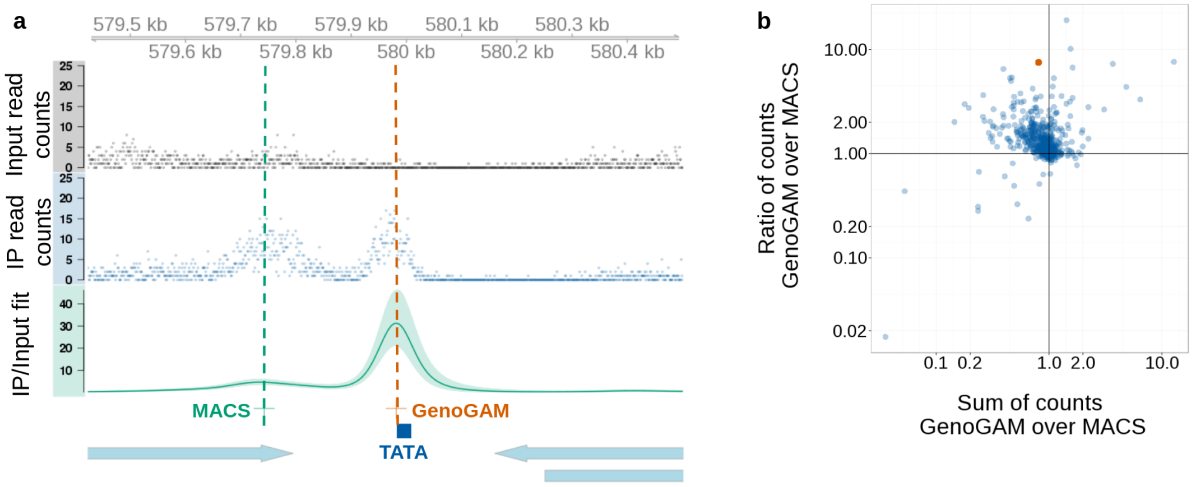

Figure 5.3: Peak positions at maximal fold change versus maximal significance. (**a**) Example region in yeast with input (black dots), IP (blue dots) and the smooth function IP over input with 95% confidence interval (green line) showing a correctly identified peak by GenoGAM (orange vertical dashed line) and an incorrect identified by MACS (green vertical dashed line), due to enrichment in input. (**b**) Scatterplot of the sum of counts (input + IP) vs ratio of counts (input/IP) for GenoGAM divided by MACS on all mutually called TATA box positions. The red dot denotes the example region shown in (a)

Next, the reason for the drastic differences observed in the yeast TFIIB dataset between GenoGAM and the other methods was investigated. The TATA-box region of *IDH2* illustrates the issue (Fig. 5.3a). The peaks reported by GenoGAM are positions with maximal a posteriori estimate of IP over input fold-changes. In contrast, MACS and JAMM report positions with maximal statistical significance [104, 145]. Because statistical significance increases with both effect size and sample size, this leads to peak calls biased toward positions with high total counts in IP and input (Fig. 5.3a). Across all 644 TATA-box regions at which both GenoGAM and MACS identify a peak, total counts within 50 bp of peak positions were higher for MACS, but count ratios were higher for GenoGAM (Fig. 5.3b), generalizing the observations made for *IDH2*.

5.1.3.2 Result on human dataset

Additionally, similar benchmark was performed on the human chromosome 22 for 6 transcription factors of the ENCODE project [79] selected to be representative of accuracies in predicting ChIP-Seq peak positions from sequence motifs [129] (CEBPB, CTCF, MAX, USF1, PAX5, and YY1). Scaling up from the small genome of S.cerevisiae to the large genome of human proved to be difficult for the first implementation of GenoGAM, due to memory consumption and long runtime. The smallest chromosome (chr22) was therefore selected for a benchmark to assess performance under low read coverage on real data.

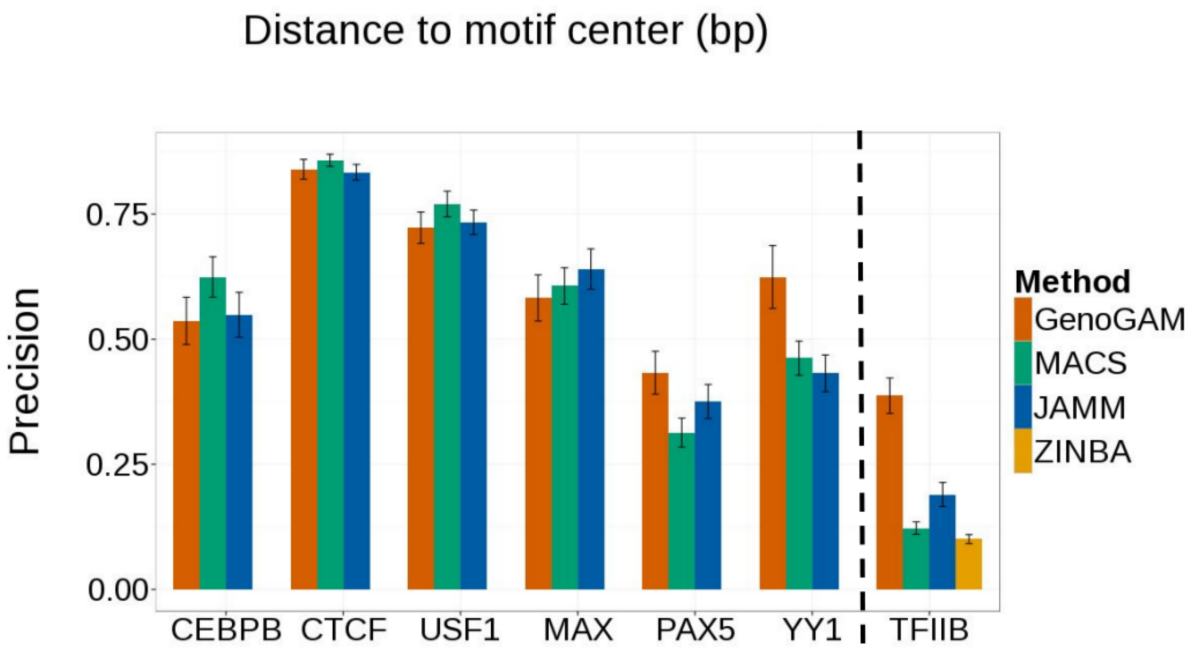

Figure 5.4: GenoGAM peak caller comparison with state-of-the-art methods on human. Proportion of significant peaks within 30 bp of motif center and 95% bootstrap confidence interval (error bars) for all six ENCODE transcription factors (CEBPB, CTCF, USF1, MAX, PAX5, YY1) on chromosome 22 and for the yeast TFIIB dataset.

On these data, GenoGAM performance was comparable to the other methods (95% boostrap confidence intervals, Fig. 5.4 for significant peaks, and Appendix Fig. B.5 for distance distributions and Appendix Fig. B.6 for all cutoffs). Hence, improvement comparable to the TFIIB benchmark could not have been repeated, but nonetheless shown to be as performant as dedicated tools. Although benchmark on yeast suggest favourable performance, benchmark on human data is of more practical value. Therefore, due to the small size of the chromosome, conclusions regarding peak calling performance on a whole human genome as well as peak calling in general should be considered with care.

The yeast TFIIB dataset was sequenced at a much higher coverage than the ENCODE dataset (0.9 unique fragments per base in average versus less than 0.03 unique fragments per base in average), leading to stronger discrepancies between significance and robust fold-changes. As sequencing depth is expected to increase in the near future, I anticipate that robust fold-change estimates as provided by GenoGAM will be a more sustainable criterion than mere significance for calling peak positions.

In order to confirm biological accuracy of GenoGAM 2.0, a replication benchmark was conducted on the same data with GenoGAM 2.0. Figure 5.5 shows that GenoGAM is on par with alternative methods to infer peak summit positions in ChIP-Seq data of transcription factors. Consistently with the fact that GenoGAM 2.0 fits the same function than GenoGAM 1.0, the performance on these two benchmarks matched.

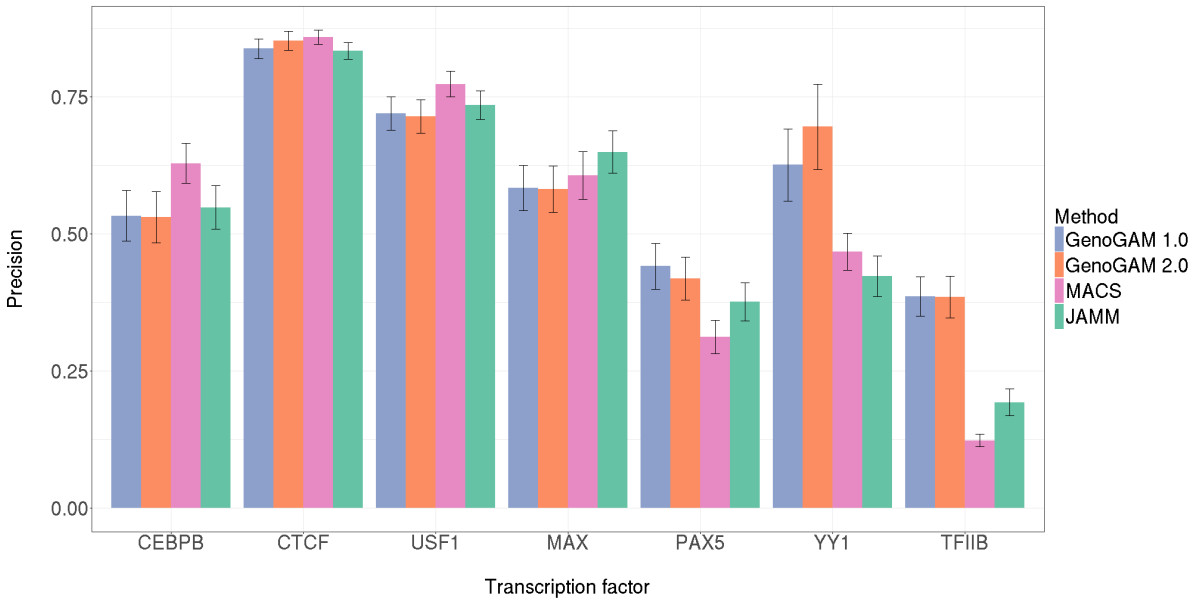

Figure 5.5: Replication of figure 5.4 with GenoGAM 2.0. Proportion of significant peaks within 30 bp of motif center and 95% bootstrap confidence interval (error bars) for six ENCODE transcription factors (CEBPB, CTCF, USF1, MAX, PAX5, YY1) on chromosome 22 and for the yeast TFIIB dataset. For simplicity, ZINBA was excluded completely.

5.2 Application to methylation

Generalized additive models are based on the generalized linear modeling framework and thus allow any distribution of the exponential family for the response. Therefore, GenoGAM can be also used to model continuous responses, for instance using the Gaussian distribution, and proportions using the Binomial distribution. For ChIP-Seq data, a log-linear predictor-response relationship of the form (Equation 3.2) is justified by the fact that effects on the mean are typically multiplicative. However, other monotonic link functions could also be used. Moreover, quasi-likelihood approaches are supported, allowing for the specification of flexible mean-variance relationships [183].

To test the flexibility of GenoGAM, a proof-of-principle study on modeling bisulfite sequencing of bulk embryonic mouse stem cells grown in serum was conducted [184]. Bisulfite sequencing quantifies methylation rate by converting cytosine residues to uracil, leaving 5-methylcytosine residues unaffected. At each cytosine, the data consisted of the number n_i of fragments overlapping the cytosine and the number y_i of these fragments for which the cytosine was not converted to uracil. The quantity of interest was the methylation rate, i.e. the expectation of the ratio y_i/n_i. In the original publication, single nucleotide position methylation rates were estimated using a sliding window approach with an ad-hoc choice of window size of 3 kb computed in steps of 600 bp. Figure 5.6 reproduces an original figure showing the fit in a 120kb section of chromosome 6. This 120 kb section was modeled with GenoGAM using a quasi-binomial model,

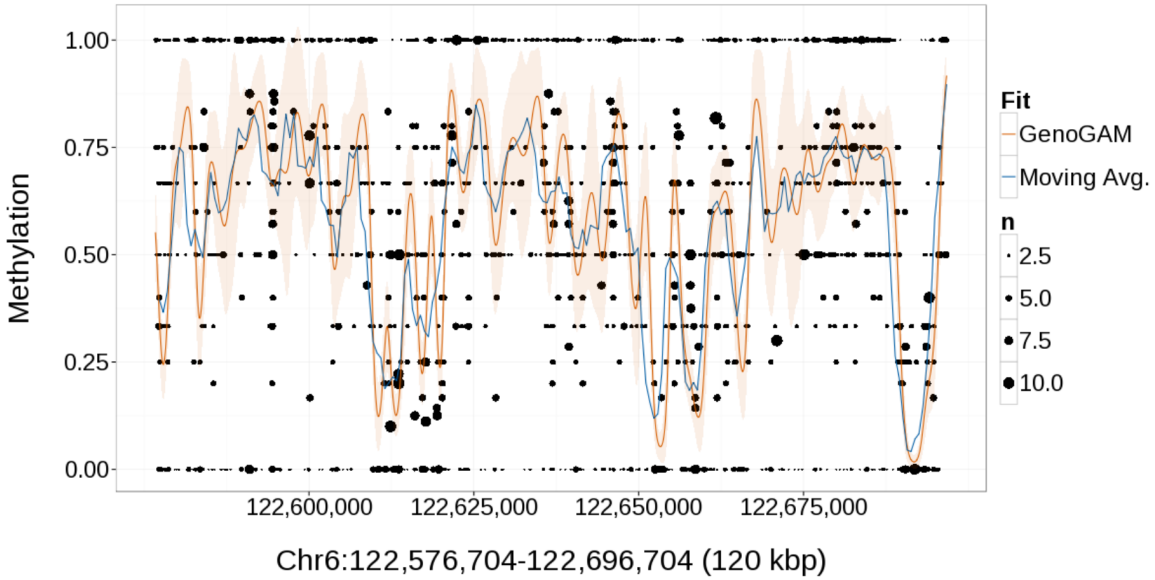

Figure 5.6: Application to DNA methylation data. Estimated DNA methylation rates in a 120 kb region of chromosome 6 of the mouse (cf. Smallwood et al.[184]). Shown are the data for bulk embryonic mouse stem cells grown in serum; ratios of methylated counts for each CpG position (black dots), with point size proportional to the number of reads. The estimated rates are shown for the moving average approach[184] of 3,000 bp bins in 600 bp steps (blue line) and for the GenoGAM (orange line) with 95% confidence band (ribbon).

where the response was the number of successes y_i out of n_i trials, the log-odd ratio was modeled as a smooth function of the genomic position, and the variance was equal to a dispersion parameter times the variance of the binomial distribution. Smoothing and dispersion parameters were determined by cross-validation. The GenoGAM fit was consistent with the original publication [184], but did not rely on manually set window sizes and provided confidence bands (Fig. 5.6). As expected, wider confidence bands were obtained in regions of sparse data and tighter bands in regions with a lot of data (Fig. 5.6).

6 Conclusion

Parts of the conclusion presented in this section is part of the manuscript "GenoGAM: genome-wide generalized additive models for ChIP-Seq analysis" from Stricker and Engelhardt et al. 2017 [1] and the manuscript "GenoGAM 2.0: Scalable and efficient implementation of genome-wide generalized additive models for gigabase-scale genomes" from Stricker et al. 2018 [2].

Advances in sequencing techniques enabled measurement of genomic information on whole genome level, such as ChIP-Seq or Bisulfite sequencing. This opened up opportunities beyond a set of well-known regions and towards the complete characterization of the physical genome. First computational methods were developed partially unaware of the underlying biases in the data with an application specific focus. Their evolution lead to more sophisticated, but also more specialized models with particular strength in a certain application, such as narrow peak calling. In particular, many were designed flexible in the parameter settings, to allow the user to tune them according to his needs. Targeting the specialization in applications and the subjectivity in hyperparameter selection allowed me to investigate a new methodology for longitudinal genome-wide data that can incorporate the strength of previous methods into a general, statistical framework.

In this thesis I have introduced this generic framework based on generalized additive models to model ChIP-Seq data. Unlike most other methods for ChIP-Seq analysis, GenoGAM is a data generative model, which gives an explicit likelihood of the data. This in turn yields an objective criterion to set the amount of smoothing. Smoothing and dispersion parameters are obtained by cross-validation, i.e. they are fitted for the accuracy in predicting unseen data. This criterion turned out to provide useful values of smoothing and dispersion for inference. Moreover it led to reasonable uncertainty estimates since confidence bands of the fits were found to be only slightly conservative. To my best knowledge, GenoGAM is the first method so far that has addressed the setting of the amount of smoothing for ChIP-Seq data. The possibility exists to estimate the smoothing and dispersion parameters separately for each sample, which would result in more robust estimates at the cost of some flexibility. However, in the analyses the samples within an experiment were all similar enough to estimate the parameters globally.

The utilization of genome-wide GAMs comes with a number of advantages: First, flexible modeling of factorial designs, as well as replicates with different sequencing depths using size factors as offsets. More elaborate usage could include position- and sample-specific copy number variations, or GC-biases. Second, applying GAMs yields confidence bands as a measure of local uncertainty for the estimated rates. I have shown how these can be the basis to compute point-wise and region-wise p-values. Third,

GAMs outputs analytically differentiable smooth functions, allowing flexible downstream analysis. I discussed how peak calling can be elegantly handled by making use of the first and second derivatives. Fourth, various link functions and distributions can be used, providing the possibility to model a wide range of genomic data beyond ChIP-Seq, as illustrated with a first application on DNA methylation. Hence, I foresee GenoGAM as a generic method for the analysis of genome-wide assays.

The method is implemented as a freely available Bioconductor `R` package `GenoGAM`. Given a configuration file of the BAM files, experiment design matrix and model formula, it will automatically estimate all parameters of the model. Alternatively, users can provide their own size factors or smoothing and overdispersion parameters. `GenoGAM` provides downstream analysis functions for differential binding and peak calling as described above. `GenoGAM` supports a number of parallel backends through the Bioconductor parallel framework `BiocParallel`.

Scalability to fit very long longitudinal data such as whole chromosomes at base-pair resolution is made possible by parallelization over the data and allowing approximations rather than exact computation of the fit [185]. Nonetheless, practical usage of the implementation of GenoGAM 1.0 is limited to organisms with small genomes such as yeast or bacteria, or to selected subsets of larger genomes, such as promoters. Improvements on computation time and memory footprint are currently available in GenoGAM 2.0 (working title "fastGenoGAM") on GitHub only and will be available in the `GenoGAM` package in the next release cycle on Bioconductor (October 2018).

GenoGAM 2.0 is a significantly improved implementation of GenoGAM 1.0 [1] on three main aspects: Data storage, parameter estimation and standard error computation. Runtime and memory footprint are shown to scale linearly with the number of parameters per tiles. As a result, GenoGAM can be applied overnight to gigabase-scale genome datasets on a typical lab server. Runtime for mega-base genomes like the yeast genome is within minutes on a standard PC. Finally, the algorithmic improvements apply to GAMs of long longitudinal data and can therefore be relevant for a broader community beyond the field of genomics.

A Appendix: Additional Methods

A.1 Differential binding data

The dataset consisted of four samples: two biological replicate IPs for the wild type strain and two biological replicate IPs for the mutant strain. Raw sequencing files:

- H3K4ME3_Full_length_Set1_Rep_1.fastq
- H3K4ME3_Full_length_Set1_Rep_2.fastq
- H3K4ME3_aa762-1080_Set1_Rep_1.fastq
- H3K4ME3_aa762-1080_Set1_Rep_2.fastq

were obtained from the Sequence Read Archive (SRA) repository (`http://www.ncbi.nlm.nih.gov/sra`). These were paired-end reads. Reads were aligned to the SacCer3 build of the *S. cerevisiae* genome with the STAR aligner [87] (version 2.4.0, default parameters). Reads with ambiguous mapping were removed using samtools [86] (version 1.2 option *-q 255*). Gene boundaries were obtained from the *S. cerevisiae* genome annotation R64.1.1, restricting gff file entries of type "gene".

A.2 Performance comparison data

These are the datasets used in the runtime and memory footprint comparison studies (section 4 and figure 4.6):

- The yeast dataset is the same as in the differential binding application (see appendix A.1).
- CEBPB: https://www.encodeproject.org/experiments/ENCSR000EHE
- FOXA1: https://www.encodeproject.org/experiments/ENCSR267DFA
- IRF9: https://www.encodeproject.org/experiments/ENCSR926KTP
- KLF1: https://www.encodeproject.org/experiments/ENCSR550HCT
- MAFG: https://www.encodeproject.org/experiments/ENCSR818DQV
- MNT: https://www.encodeproject.org/experiments/ENCSR261EDU

- NRF1: https://www.encodeproject.org/experiments/ENCSR135ANT

Human datasets were downloaded from the ENCODE project website, the IDs for each dataset are the last parts of the URLs. Reads were aligned to the hg38 build of the human genome with the STAR aligner [87] (version 2.4.0, default parameters). Reads with ambiguous mapping were removed using samtools [86] (version 1.2 option *-q 255*).

A.3 Peak calling

A.3.1 Yeast TFIIB ChIP-Seq dataset

ChIP-Seq for TFIIB was performed essentially as described previously [186] with a few modifications. Briefly, 600 ml BY 4741 S. cerevisiae culture with C-terminally TAP-tagged TFIIB (Open Biosystems) was used. Immunoprecipitation was performed with 75 μl of IgG SepharoseTM 6 Fast Flow beads (GE Healthcare) for 3 hours at 4°C on a turning wheel. 30 μl of Input sample was taken before immunoprecipitation and stored at 4°C. IP and Input samples were reverse cross-linking for 2 hours with Proteinase K at 65°C and purified using Quiagen MinElute Kit. Samples were digested with 2.5 μl RNase A/T1 Mix (2 mg/ml RNase A, 5000 U/ml RNase T1; Fermentas) at 37°C for 1 h, purified and eluted in 50 μl H2O. ChIP-Seq libraries were prepared using NEB Next library preparation kit following manufacturer's instructions using the complete 50 μl as input. 2 μl of 1.7 μM adapters containing a GGAT barcode and 2 μl of a 0.25 μM adapter containing a CACT barcode were used for ligation with Input and IP samples, respectively. The final library was amplified for 22 cycles using Phusion Polymerase and purified using Agencourt Magnetic beads. 36 bp single end sequencing was performed on an Illumina GAIIX sequencer at the LAFUGA core facility of the Gene Center, Munich. Single-end 36 base reads and 4 base reads of barcodes were obtained and processed using the Galaxy platform [187]. Reads were demultiplexed, quality-trimmed (Fastq Quality Filter), and mapped with Bowtie 0.12.7 [188] to the SacCer2 genome assembly (Bowtie options: -q -p 4 -S –sam-nohead -phred33-quals). ChIP-Seq data are available at Array Express under the accession number E-MTAB-4175.

A.3.2 The data

The TFIIB yeast dataset consisted of two samples: one input and one IP without replicates. For about 20% of yeast promoters, recruitment of TFIIB is triggered by the well characterized DNA element TATA-box, providing at these promoters a ground truth for a TFIIB occupancy peak [182]. Location of the TATA-boxes where defined as instances of the motif `TATAWAWR` [182] at most 200 bp 5' and 50 bp 3' of one of the 7,272 transcript 5'-ends reported by Xu et al. (2009) [165]. In total 1,105 TATA-boxes were mapped genome-wide by regular expression of the consensus motif.

For the human dataset alignment files (BAM files, aligned for the human genome assembly hg19) for ChIP-Seq data for the transcription factors CEBPB, CTCF, MAX, USF1, PAX5, and YY1 were obtained from the ENCODE website `www.encodeproject.`

org. All these datasets contained two biological replicates for the protein samples and at least one input sample. However, the library sizes of the input samples were so low that including them resulted in higher uncertainty about the peaks. Therefore analyses was conducted without correction for input. Motif occurrences in the genome were determined by FIMO [189] using default threshold 10^{-4} with position weight matrices (PWMs) from the JASPAR 2014 database [190] with the following IDs: CEBPB: MA0466.1, CTCF: MA0139.1, MAX: MA0058.1, PAX5: MA0014.2, USF1: MA0093.2, YY1: MA0095.2

A.3.3 Method specification

The version 2 of the MACS software, MACS2, was run with the default parameters and the additional flag *call-summits*. In case of TFIIB, the *nomodel* parameter was used to avoid building the shifting model. This was necessary since the default values for *mfold* were too high and resulted in worse performance if reduced, compared to absence of a model.

JAMM was run with default values and peak calling mode (*-m*) set to narrow assuming a three component mixture model for background, enriched regions and tails of enriched regions. Although JAMM computes a score to rank peaks it does not provide a method to define a threshold for a given FDR or significance. Nevertheless, JAMM applies some filtering on the complete list of peaks to output a filtered list. Instead of using this filtered output directly, the complete, sorted (by score) peak list was used and the top N results where taken, where N is the number of peaks in the filtered output. This improved the performance of JAMM in some cases (and left unchanged in others). For analysis, where a cutoff for JAMM was still needed the same number of peaks that MACS reported was used.

For ZINBA, the mappability score was generated (*generateAlignability*) with the mappability files for 36 bp reads, taken from the ZINBA website `https://code.google.com/p/zinba/`. The average fragment length (*extension*) was specified at 190 bp, window size (*winSize*) at 250 and offset (*offset*) at 125. The FDR threshold was set to 0.1 and window gap to 0. Peaks were refined (default) and model selection was activated. The complete model was used (*selecttype = "complete"*), input was included as a covariate (*selectcovs = "input_count"*) and interactions were allowed. The chromosome used to build the model was selected randomly to be "chrXVI" (*selectchr*). The parameter "method" was set *method = "mixture"*. During application to human chromosome 22, ZINBA was very unstable resulting in errors. It was therefore excluded from analysis on human data, but kept for yeast.

B Appendix: Additional Figures

B.1 Differential binding

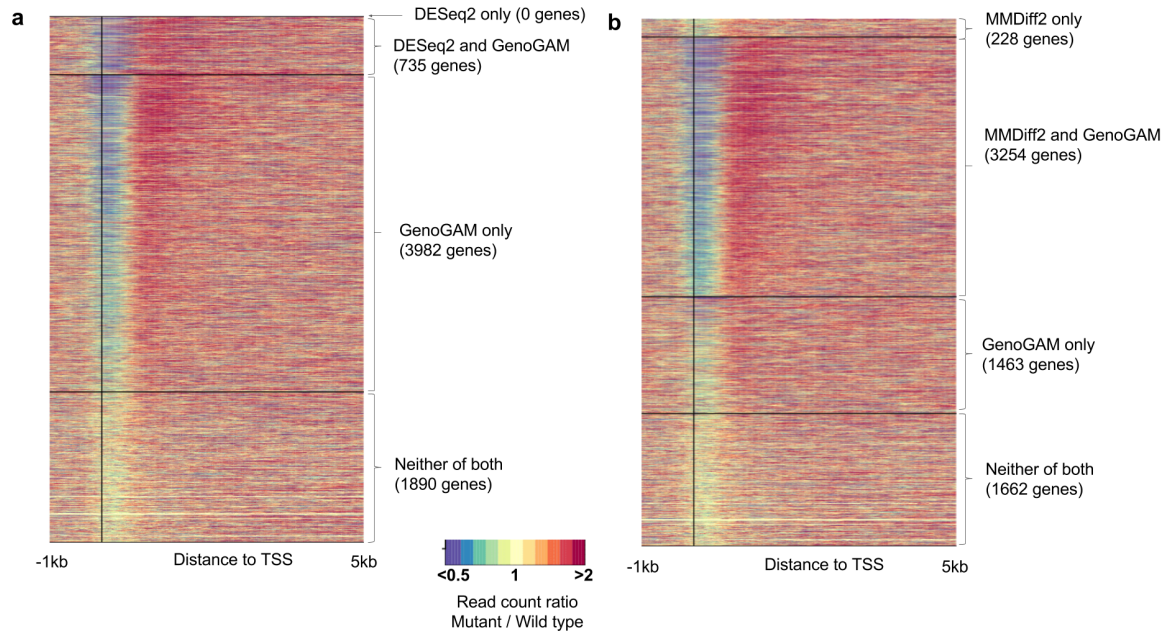

Figure B.1: Segmented gene heatmap: DESeq2 and MMDiff2 Read count ratio of all 6607 genes (y-axis) centered on the TSS. The genes are segmented into four groups: Significant genes called only by the competitor method, called by competitor method and GenoGAM, only GenoGAM and neither of both (top to bottom). Within each segment the genes are ordered by p-value (top is the lowest and bottom the highest) of the respective method or in case of both methods present: GenoGAM. (**a**) For competitor method DESeq2. The number of the respective genes is stated in brackets. All DESeq2 genes are a subset of GenoGAM genes. (**b**) For competitor method MMDiff2. The number of the respective genes is stated in brackets. Both methods seem to agree on the visually most significant genes (clear depletion around TSS and increase in the gene body)

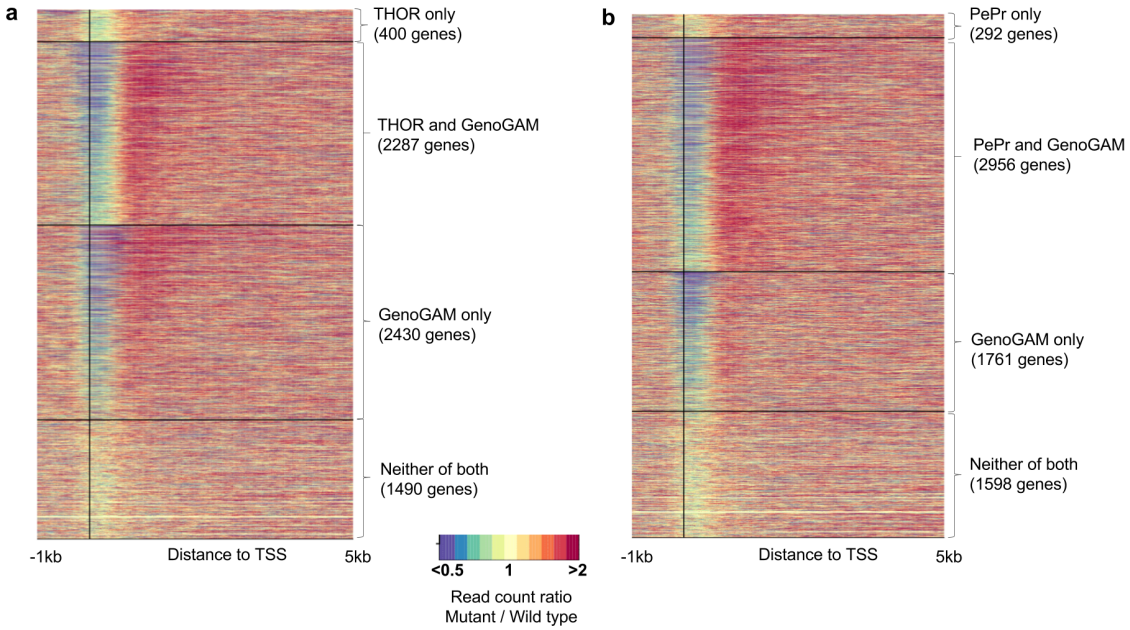

Figure B.2: Segmented gene heatmap: THOR and PePr Read count ratio of all 6607 genes (y-axis) centered on the TSS. The genes are segmented into four groups: Significant genes called only by the competitor method, called by competitor method and GenoGAM, only GenoGAM and neither of both (top to bottom). Within each segment the genes are ordered by p-value (top is the lowest and bottom the highest) of the respective method or in case of both methods present: GenoGAM. (**a**) For competitor method THOR. The number of the respective genes is stated in brackets. Whereas many common called genes seem to be visually valid, THOR misses a large number of strongly differentially bound genes, while calling some with a weaker signal (**b**) For competitor method PePr. The number of the respective genes is stated in brackets. Whereas many common called genes seem to be visually valid, PePr misses a large number of strongly differentially bound genes, while calling some with a weaker signal.

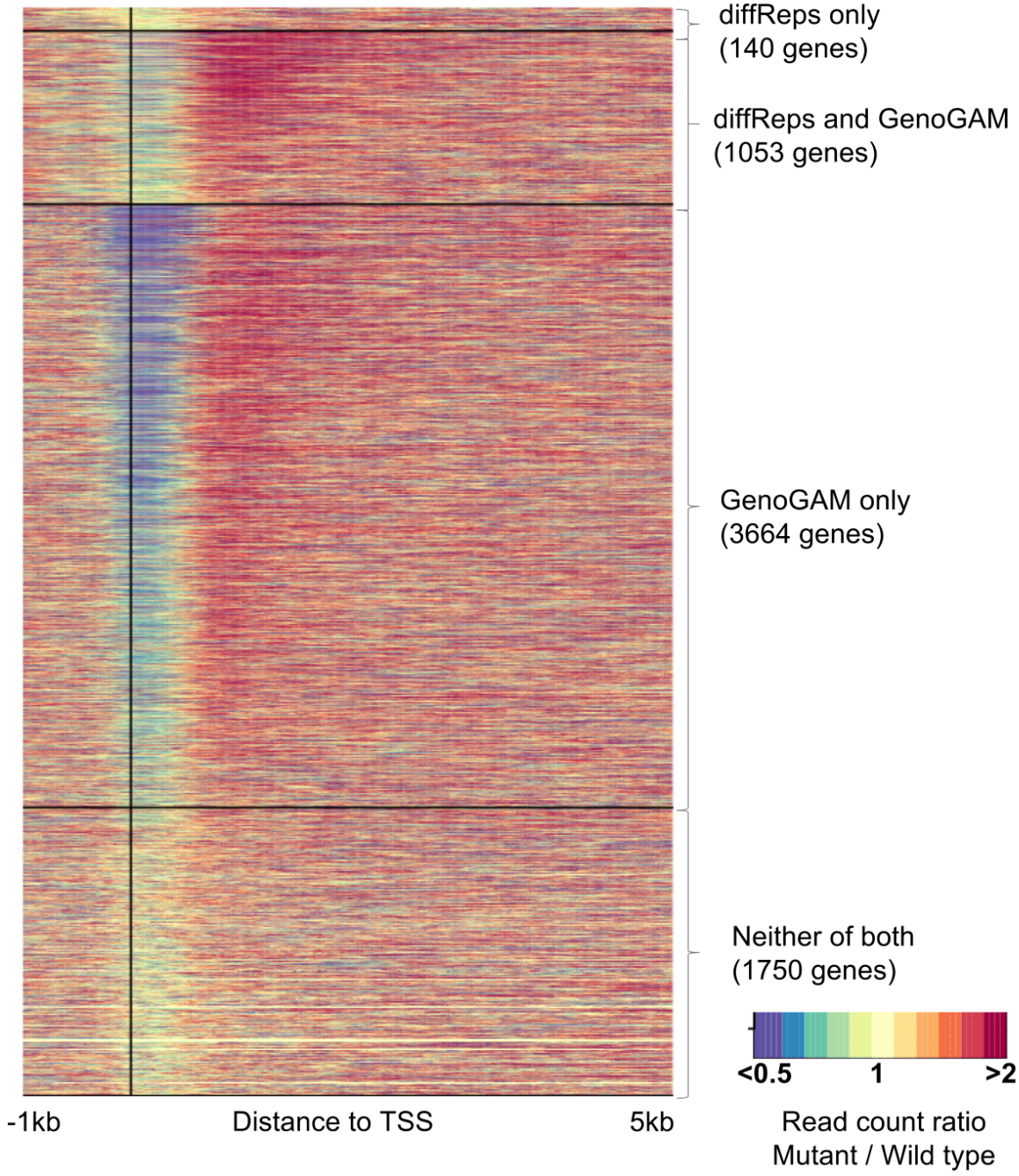

Figure B.3: Segmented gene heatmap: diffReps Read count ratio of all 6607 genes (y-axis) centered on the TSS. The genes are segmented into four groups: Significant genes called only by the competitor method, called by competitor method and GenoGAM, only GenoGAM and neither of both (top to bottom). Within each segment the genes are ordered by p-value (top is the lowest and bottom the highest) of the respective method or in case of both methods present: GenoGAM. The number of the respective genes is stated in brackets. Whereas many common called genes seem to be visually valid, they seem nonetheless to be less differentially bound then a large number of genes called by GenoGAM. The number of genes called by diffReps alone seem to reflect this even more.

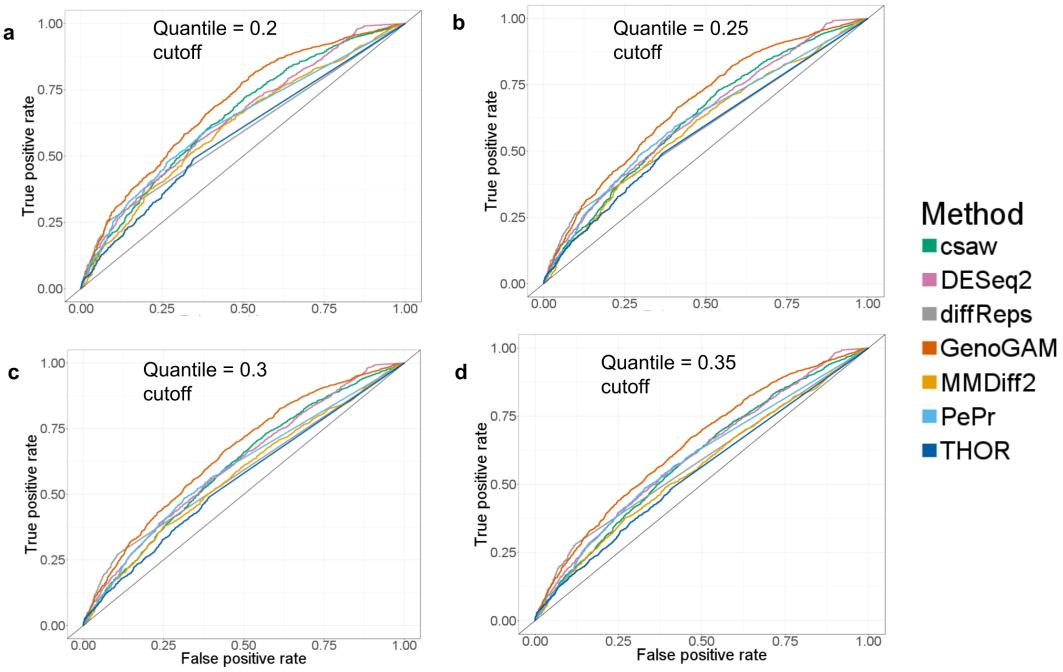

Figure B.4: ROC curves Further ROC curves as in figure 3.10 with quantile cutoffs of 0.2 (top left), 0.25 (top right), 0.3 (bottom left) and 0.35 (bottom right). In all plots GenoGAM shows consistently a higher True positive rate (TPR) while maintaining a lower False positive rate (FPR).

B.2 Peak calling

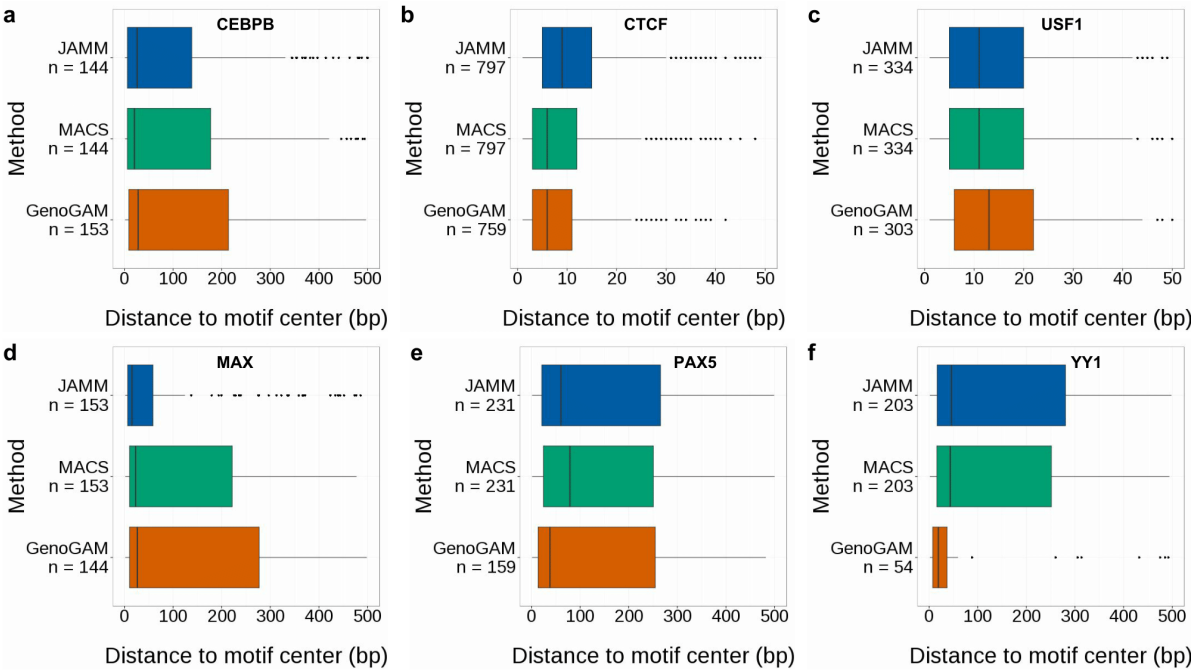

Figure B.5: Distances to motif center for ENCODE data. As in Figure 5.2a for the ENCODE transcription factors CEBPP (**a**), CTCF(**b**), USF1 (**c**), MAX (**d**), PAX5 (**e**), and YY1 (**f**).

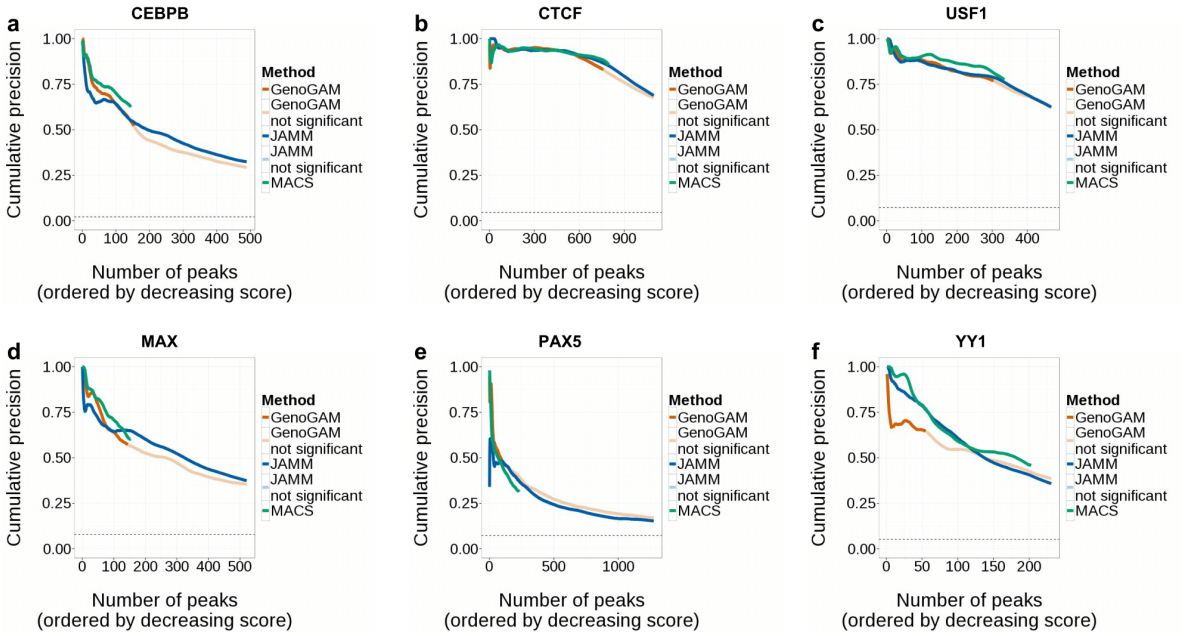

Figure B.6: Proportion of peaks for ENCODE data. As in Figure 5.2b for the ENCODE transcription factors CEBPP (**a**), CTCF(**b**), USF1 (**c**), MAX (**d**), PAX5 (**e**), and YY1 (**f**).

List of Figures

List of Tables

Acronyms

AUC	area under the ROC curve.
BAM	Binary Alignment Map.
ChAP	chromatin immunoprecipitation assay with arbitrarily primed PCR.
ChIP	chromatin immunoprecipitation.
ChIP-chip	chromatin immunoprecipitation with DNA microarray.
ChIP-exo	chromatin immunoprecipitation with massively parallel DNA sequencing and exonuclease treatment.
ChIP-nexus	chromatin immunoprecipitation with massively parallel DNA sequencing, exonuclease treatment, unique barcode and single ligation.
ChIP-PET	chromatin immunoprecipitation with the paired-end ditag.
ChIP-Seq	chromatin immunoprecipitation with massively parallel DNA sequencing.
Dam	DNA adenine methyltransferase.
DamID	DNA adenine methyltransferase identification.
DNA	Deoxyribonucleic acid.
FDR	False Discovery Rate.
FPR	False positive rate.
GAM	generalized additive model.
GCV	Generalized Cross Validation.
GenoGAM	genome-wide generalized additive model.
GLM	generalized linear model.
GMAT	genome-wide mapping technique.
H3K4me3	histone H3 Lysine 4 trimethylation.
HDF5	Hierarchical Data Format.
HMM	Hidden Markov Models.

IDR	Irreproducible Discovery Rate.
IP	immunoprecipitation.
IRLS	Iteratively Reweighted Least Squares.
LOESS	Local Regression.
MAD	median absolute deviation.
MMD	Maximum Mean Discrepancy.
MS-PET	multiplex sequencing with paired-end ditagging.
NGS	next-generation sequencing.
P-IRLS	Penalized Iteratively Reweighted Least Squares.
P-Spline	Penalized B-splines.
PCR	Polymerase chain reaction.
PIC	preinitiation complex.
Pol II	RNA polymerase II.
PWMs	position weight matrices.
RNA	Ribonucleic acid.
ROC	receiver operating characteristic.
SACO	serial analysis of chromatin occupancy.
STAGE	sequence tag analysis of genomic enrichment.
TF	transcription factor.
TFBS	transcription factor binding site.
TPR	True positive rate.
TSS	transcription start site.

References

[1] Stricker, G. *et al.* GenoGAM: Genome-wide generalized additive models for ChIP-Seq analysis. *Bioinformatics* **33** (2017).

[2] Stricker, G., Galinier, M. & Gagneur, J. GenoGAM 2.0: Scalable and efficient implementation of genome-wide generalized additive models for gigabase-scale genomes. *BMC Bioinformatics* **19**, 247 (2018). URL `https://doi.org/10.1186/s12859-018-2238-7`.

[3] Sanger, F. *et al.* Nucleotide sequence of bacteriophage ϕX174 DNA. *Nature* **265**, 687 (1977). URL `http://dx.doi.org/10.1038/265687a0`.

[4] International Human Genome Sequencing Consortium. Finishing the euchromatic sequence of the human genome. *Nature* **431**, 931 (2004). URL `http://dx.doi.org/10.1038/nature03001`.

[5] The Yeast Genome Directory. The Yeast Genome Directory. *Nature* **387** (1997).

[6] The C. elegans Sequencing Consortium. Genome sequence of the nematode C. elegans: A platform for investigating biology. *Science* (1998).

[7] Adams, M. D. *et al.* The Genome Sequence of Drosophila melanogaster. *Science* **287**, 2185 (2000). URL `http://science.sciencemag.org/content/287/5461/2185.abstract`.

[8] Mouse Genome Sequencing Consortium. Initial sequencing and comparative analysis of the mouse genome. *Nature* **420**, 520 (2002). URL `http://dx.doi.org/10.1038/nature01262`.

[9] The ENCODE Project Consortium. The ENCODE (ENCyclopedia of DNA Elements) Project (2004). `arXiv:1011.1669v3`.

[10] Latchman, D. S. Transcription factors: An overview. *The International Journal of Biochemistry & Cell Biology* **29**, 1305–1312 (1997). URL `https://www.sciencedirect.com/science/article/pii/S135727259700085X?via{%}3Dihub`.

[11] Reid, J. L., Iyer, V. R., Brown, P. O. & Struhl, K. Coordinate regulation of yeast ribosomal protein genes is associated with targeted recruitment of Esa1 histone acetylase. *Molecular cell* **6**, 1297–1307 (2000). URL `https://doi.org/10.1016/S1097-2765(00)00128-3`.

[12] Ren, B. *et al.* Genome-Wide Location and Function of DNA Binding Proteins. *Science* **290**, 2306 (2000). URL http://science.sciencemag.org/content/290/5500/2306.abstract.

[13] Weinmann, A. S., Yan, P. S., Oberley, M. J., Huang, T. H.-M. & Farnham, P. J. Isolating human transcription factor targets by coupling chromatin immunoprecipitation and CpG island microarray analysis. *Genes & Development* **16**, 235–244 (2002). URL http://www.ncbi.nlm.nih.gov/pmc/articles/PMC155318/.

[14] Hug, B. A., Ahmed, N., Robbins, J. A. & Lazar, M. A. A Chromatin Immunoprecipitation Screen Reveals Protein Kinase Cβ as a Direct RUNX1 Target Gene. *Journal of Biological Chemistry* **279**, 825–830 (2004). URL http://www.jbc.org/content/279/2/825.abstract.

[15] Zeller, K. I. *et al.* Global mapping of c-Myc binding sites and target gene networks in human B cells. *Proceedings of the National Academy of Sciences of the United States of America* **103**, 17834–17839 (2006). URL http://www.ncbi.nlm.nih.gov/pmc/articles/PMC1635161/.

[16] Lieb, J. D., Liu, X., Botstein, D. & Brown, P. O. Promoter-specific binding of Rap1 revealed by genome-wide maps of protein–DNA association. *Nature Genetics* **28**, 327 (2001). URL http://dx.doi.org/10.1038/ng569.

[17] Blackwood, E. M. & Kadonaga, J. T. Going the Distance: A Current View of Enhancer Action. *Science* **281**, 60 (1998). URL http://science.sciencemag.org/content/281/5373/60.abstract.

[18] Maston, G. A., Evans, S. K. & Green, M. R. Transcriptional Regulatory Elements in the Human Genome. *Annual Review of Genomics and Human Genetics* **7**, 29–59 (2006). URL https://doi.org/10.1146/annurev.genom.7.080505.115623.

[19] Pennacchio, L. A., Bickmore, W., Dean, A., Nobrega, M. A. & Bejerano, G. Enhancers: five essential questions. *Nature reviews. Genetics* **14**, 288–295 (2013). URL http://www.ncbi.nlm.nih.gov/pmc/articles/PMC4445073/.

[20] Kulaeva, O. I., Nizovtseva, E. V., Polikanov, Y. S., Ulianov, S. V. & Studitsky, V. M. Distant Activation of Transcription: Mechanisms of Enhancer Action. *Molecular and Cellular Biology* **32**, 4892–4897 (2012). URL http://mcb.asm.org/content/32/24/4892.abstract.

[21] Iyer, V. R. *et al.* Genomic binding sites of the yeast cell-cycle transcription factors SBF and MBF. *Nature* **409**, 533 (2001). URL http://dx.doi.org/10.1038/35054095.

[22] Martone, R. *et al.* Distribution of NF-κB-binding sites across human chromosome 22. *Proceedings of the National Academy of Sciences* **100**, 12247 (2003). URL http://www.pnas.org/content/100/21/12247.abstract.

[23] Bulyk, M. L., Huang, X., Choo, Y. & Church, G. M. Exploring the DNA-binding specificities of zinc fingers with DNA microarrays. *Proceedings of the National Academy of Sciences of the United States of America* **98**, 7158–7163 (2001). URL http://www.ncbi.nlm.nih.gov/pmc/articles/PMC34639/.

[24] Kurdistani, S. K. & Grunstein, M. Histone acetylation and deacetylation in yeast. *Nature Reviews Molecular Cell Biology* **4**, 276 (2003). URL http://dx.doi.org/10.1038/nrm1075.

[25] Kurdistani, S. K., Tavazoie, S. & Grunstein, M. Mapping Global Histone Acetylation Patterns to Gene Expression. *Cell* **117**, 721–733 (2004). URL http://dx.doi.org/10.1016/j.cell.2004.05.023.

[26] Pollack, J. R. & Iyer, V. R. Characterizing the physical genome. *Nature Genetics* **32**, 515 (2002). URL http://dx.doi.org/10.1038/ng1035.

[27] Lee, T. I. *et al.* Transcriptional Regulatory Networks in Saccharomyces cerevisiae. *Science* **298**, 799 (2002). URL http://science.sciencemag.org/content/298/5594/799.abstract.

[28] Yu, H., Luscombe, N. M., Qian, J. & Gerstein, M. Genomic analysis of gene expression relationships in transcriptional regulatory networks. *Trends in Genetics* **19**, 422–427 (2003). URL http://dx.doi.org/10.1016/S0168-9525(03)00175-6.

[29] Mirza, A. *et al.* Global transcriptional program of p53 target genes during the process of apoptosis and cell cycle progression. *Oncogene* **22**, 3645 (2003). URL http://dx.doi.org/10.1038/sj.onc.1206477.

[30] Loh, Y.-H. *et al.* The Oct4 and Nanog transcription network regulates pluripotency in mouse embryonic stem cells. *Nature Genetics* **38**, 431 (2006). URL http://dx.doi.org/10.1038/ng1760.

[31] Elgin, S. C. R. & Weintraub, H. Chromosomal Proteins and Chromatin Structure. *Annual Review of Biochemistry* **44**, 725–774 (1975). URL https://doi.org/10.1146/annurev.bi.44.070175.003453.

[32] Turner, B. M. Histone acetylation and an epigenetic code. *BioEssays* **22**, 836–845 (2000).

[33] Strahl, B. D. & Allis, C. D. The language of covalent histone modifications. *Nature* **403**, 41 (2000). URL http://dx.doi.org/10.1038/47412.

[34] Schreiber, S. L. & Bernstein, B. E. Signaling Network Model of Chromatin. *Cell* **111**, 771–778 (2002). URL http://dx.doi.org/10.1016/S0092-8674(02)01196-0.

[35] Henikoff, S. Histone modifications: Combinatorial complexity or cumulative simplicity? *Proceedings of the National Academy of Sciences of the United States of America* **102**, 5308–5309 (2005). URL http://www.ncbi.nlm.nih.gov/pmc/articles/PMC556254/.

[36] Berger, S. L. Histone modifications in transcriptional regulation. *Current Opinion in Genetics & Development* **12**, 142–148 (2002). URL https://www.sciencedirect.com/science/article/pii/S0959437X02002794?via{%}3Dihub.

[37] Wu, J. & Grunstein, M. 25 years after the nucleosome model: chromatin modifications. *Trends in Biochemical Sciences* **25**, 619–623 (2000). URL http://dx.doi.org/10.1016/S0968-0004(00)01718-7.

[38] Zhang, Y. & Reinberg, D. Transcription regulation by histone methylation: interplay between different covalent modifications of the core histone tails. *Genes & Development* **15**, 2343–2360 (2001). URL http://genesdev.cshlp.org/content/15/18/2343.short.

[39] Nielsen, S. J. *et al.* Rb targets histone H3 methylation and HP1 to promoters. *Nature* **412**, 561 (2001). URL http://dx.doi.org/10.1038/35087620.

[40] Saunders, A. *et al.* Tracking FACT and the RNA Polymerase II Elongation Complex Through Chromatin in Vivo. *Science* **301**, 1094 (2003). URL http://science.sciencemag.org/content/301/5636/1094.abstract.

[41] Roh, T.-Y., Cuddapah, S., Cui, K. & Zhao, K. The genomic landscape of histone modifications in human T cells. *Proceedings of the National Academy of Sciences of the United States of America* **103**, 15782–15787 (2006). URL http://www.ncbi.nlm.nih.gov/pmc/articles/PMC1613230/.

[42] Soutourina, J. Transcription regulation by the Mediator complex. *Nature Reviews Molecular Cell Biology* **19**, 262 (2017). URL https://www.nature.com/articles/nrm.2017.115.

[43] Zemach, A., McDaniel, I. E., Silva, P. & Zilberman, D. Genome-Wide Evolutionary Analysis of Eukaryotic DNA Methylation. *Science* **328**, 916 (2010). URL http://science.sciencemag.org/content/328/5980/916.abstract.

[44] Jones, P. A. & Laird, P. W. Cancer-epigenetics comes of age. *Nature Genetics* **21**, 163 (1999). URL http://dx.doi.org/10.1038/5947.

[45] Wu, T. P. *et al.* DNA methylation on N(6)-adenine in mammalian embryonic stem cells. *Nature* **532**, 329–333 (2016). URL http://www.ncbi.nlm.nih.gov/pmc/articles/PMC4977844/.

[46] Bird, A. P. CpG-rich islands and the function of DNA methylation. *Nature* **321**, 209 (1986). URL http://dx.doi.org/10.1038/321209a0.

[47] Du, J., Johnson, L. M., Jacobsen, S. E. & Patel, D. J. DNA methylation pathways and their crosstalk with histone methylation. *Nature Reviews Molecular Cell Biology* (2015).

[48] Axelrod, J. D. & Majors, J. An improved method for photofootprinting yeast genes in vivo using Taq polymerase. *Nucleic Acids Research* **17**, 171–183 (1989). URL `http://www.ncbi.nlm.nih.gov/pmc/articles/PMC331543/`.

[49] Sudarsanam, P., Iyer, V. R., Brown, P. O. & Winston, F. Whole-genome expression analysis of snf/swi mutants of Saccharomyces cerevisiae. *Proceedings of the National Academy of Sciences of the United States of America* **97**, 3364–3369 (2000). URL `http://www.ncbi.nlm.nih.gov/pmc/articles/PMC16245/`.

[50] DeRisi, J. L., Iyer, V. R. & Brown, P. O. Exploring the Metabolic and Genetic Control of Gene Expression on a Genomic Scale. *Science* **278**, 680 (1997). URL `http://science.sciencemag.org/content/278/5338/680.abstract`.

[51] Chu, S. *et al.* The Transcriptional Program of Sporulation in Budding Yeast. *Science* **282**, 699 (1998). URL `http://science.sciencemag.org/content/282/5389/699.abstract`.

[52] Horak, C. E. & Snyder, M. ChIP-chip: A genomic approach for identifying transcription factor binding sites. *Methods in Enzymology* **350**, 469–483 (2002). URL `https://www.sciencedirect.com/science/article/pii/S0076687902509794`.

[53] Solomon, M. J. & Varshavsky, A. Formaldehyde-mediated DNA-protein crosslinking: a probe for in vivo chromatin structures. *Proceedings of the National Academy of Sciences of the United States of America* **82**, 6470–6474 (1985). URL `http://www.ncbi.nlm.nih.gov/pmc/articles/PMC390738/`.

[54] McGhee, J. D. & Von Hippel, P. H. Formaldehyde as a probe of DNA structure. I. Reaction with exocyclic amino groups of DNA bases. *Biochemistry* **14**, 1281–1296 (1975). URL `https://doi.org/10.1021/bi00677a029`.

[55] Solomon, M. J., Larsen, P. L. & Varshavsky, A. Mapping proteinDNA interactions in vivo with formaldehyde: Evidence that histone H4 is retained on a highly transcribed gene. *Cell* **53**, 937–947 (1988). URL `http://dx.doi.org/10.1016/S0092-8674(88)90469-2`.

[56] Dedon, P. C., Soults, J. A., Allis, C. D. & Gorovsky, M. A. Formaldehyde crosslinking and immunoprecipitation demonstrate developmental changes in H1 association with transcriptionally active genes. *Molecular and Cellular Biology* **11**, 1729–1733 (1991). URL `http://www.ncbi.nlm.nih.gov/pmc/articles/PMC369483/`.

[57] Dedon, P. C., Soults, J. A., David Allis, C. & Gorovsky, M. A. A simplified formaldehyde fixation and immunoprecipitation technique for studying protein-DNA interactions. *Analytical Biochemistry* **197**, 83–90 (1991). URL `https://www.sciencedirect.com/science/article/pii/0003269791903592`.

[58] Orlando, V. & Paro, R. Mapping polycomb-repressed domains in the bithorax complex using in vivo formaldehyde cross-linked chromatin. *Cell* **75**, 1187–1198 (1993). URL https://www.sciencedirect.com/science/article/pii/009286749390328N?{_}rdoc=1{&}{_}fmt=high{&}{_}origin=gateway{&}{_}docanchor={&}md5=b8429449ccfc9c30159a5f9aeaa92ffb.

[59] Orlando, V., Strutt, H. & Paro, R. Analysis of Chromatin Structure byin VivoFormaldehyde Cross-Linking. *Methods* **11**, 205–214 (1997). URL https://www.sciencedirect.com/science/article/pii/S1046202396904077?via{%}3Dihub.

[60] Strutt, H., Cavalli, G. & Paro, R. Co-localization of Polycomb protein and GAGA factor on regulatory elements responsible for the maintenance of homeotic gene expression. *The EMBO Journal* **16**, 3621–3632 (1997). URL http://dx.doi.org/10.1093/emboj/16.12.3621.

[61] Hecht, A., Strahl-Bolsinger, S. & Grunstein, M. Spreading of transcriptional represser SIR3 from telomeric heterochromatin. *Nature* **383**, 92 (1996). URL http://dx.doi.org/10.1038/383092a0.

[62] Strahl-Bolsinger, S., Hecht, A., Luo, K. & Grunstein, M. SIR2 and SIR4 interactions differ in core and extended telomeric heterochromatin in yeast. *Genes & Development* **11**, 83–93 (1997). URL http://genesdev.cshlp.org/content/11/1/83.abstract.

[63] van Steensel, B. & Henikoff, S. Identification of in vivo DNA targets of chromatin proteins using tethered Dam methyltransferase. *Nature Biotechnology* **18**, 424 (2000). URL http://dx.doi.org/10.1038/74487.

[64] Orian, A. Chromatin profiling, DamID and the emerging landscape of gene expression. *Current opinion in genetics & development* **16**, 157–164 (2006). URL https://doi.org/10.1016/j.gde.2006.02.008.

[65] Roh, T.-y., Ngau, W. C., Cui, K., Landsman, D. & Zhao, K. High-resolution genome-wide mapping of histone modifications. *Nature Biotechnology* **22**, 1013 (2004). URL http://dx.doi.org/10.1038/nbt990.

[66] Liang, G. *et al.* Distinct localization of histone H3 acetylation and H3-K4 methylation to the transcription start sites in the human genome. *Proceedings of the National Academy of Sciences of the United States of America* **101**, 7357–7362 (2004). URL http://www.ncbi.nlm.nih.gov/pmc/articles/PMC409923/.

[67] Impey, S. *et al.* Defining the CREB Regulon: A Genome-Wide Analysis of Transcription Factor Regulatory Regions. *Cell* **119**, 1041–1054 (2004). URL http://dx.doi.org/10.1016/j.cell.2004.10.032.

[68] Kim, J., Bhinge, A. A., Morgan, X. C. & Iyer, V. R. Mapping DNA-protein interactions in large genomes by sequence tag analysis of genomic enrichment. *Nature Methods* **2**, 47 (2005). URL http://dx.doi.org/10.1038/nmeth726http://dx.doi.org/10.1038/nmeth726http://dx.doi.org/10.1038/nmeth726.

[69] Wei, C.-L. *et al.* A Global Map of p53 Transcription-Factor Binding Sites in the Human Genome. *Cell* **124**, 207–219 (2006). URL http://dx.doi.org/10.1016/j.cell.2005.10.043.

[70] Ng, P. *et al.* Multiplex sequencing of paired-end ditags (MS-PET): a strategy for the ultra-high-throughput analysis of transcriptomes and genomes. *Nucleic Acids Research* **34**, e84–e84 (2006). URL http://www.ncbi.nlm.nih.gov/pmc/articles/PMC1524903/.

[71] Schones, D. E. & Zhao, K. Genome-wide approaches to studying chromatin modifications. *Nature Reviews Genetics* **9**, 179 (2008). URL http://dx.doi.org/10.1038/nrg2270.

[72] Albert, I. *et al.* Translational and rotational settings of H2A.Z nucleosomes across the Saccharomyces cerevisiae genome. *Nature* **446**, 572 (2007). URL http://dx.doi.org/10.1038/nature05632.

[73] Barski, A. *et al.* High-Resolution Profiling of Histone Methylations in the Human Genome. *Cell* **129**, 823–837 (2007). URL http://dx.doi.org/10.1016/j.cell.2007.05.009.

[74] Mikkelsen, T. S. *et al.* Genome-wide maps of chromatin state in pluripotent and lineage-committed cells. *Nature* **448**, 553–560 (2007). URL http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2921165/.

[75] Johnson, D. S., Mortazavi, A., Myers, R. M. & Wold, B. Genome-Wide Mapping of in Vivo Protein-DNA Interactions. *Science* **316**, 1497 (2007). URL http://science.sciencemag.org/content/316/5830/1497.abstract.

[76] Robertson, G. *et al.* Genome-wide profiles of STAT1 DNA association using chromatin immunoprecipitation and massively parallel sequencing. *Nature Methods* **4**, 651 (2007). URL http://dx.doi.org/10.1038/nmeth1068.

[77] Quail, M. A. *et al.* A large genome center's improvements to the Illumina sequencing system. *Nature Methods* **5**, 1005 (2008). URL http://dx.doi.org/10.1038/nmeth.1270http://dx.doi.org/10.1038/nmeth.1270http://dx.doi.org/10.1038/nmeth.1270.

[78] Park, P. J. ChIP–seq: advantages and challenges of a maturing technology. *Nature Reviews Genetics* **10**, 669 (2009). URL http://dx.doi.org/10.1038/nrg2641.

[79] Landt, S. G. *et al.* ChIP-seq guidelines and practices of the ENCODE and modENCODE consortia. *Genome Research* **22**, 1813–1831 (2012). URL `http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3431496/`.

[80] Peng, S., Alekseyenko, A. A., Larschan, E., Kuroda, M. I. & Park, P. J. Normalization and experimental design for ChIP-chip data. *BMC Bioinformatics* **8**, 219 (2007). URL `https://doi.org/10.1186/1471-2105-8-219`.

[81] Rhee, H. S. & Pugh, B. F. Comprehensive Genome-wide Protein-DNA Interactions Detected at Single Nucleotide Resolution. *Cell* **147**, 1408–1419 (2011). URL `http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3243364/`.

[82] He, Q., Johnston, J. & Zeitlinger, J. ChIP-nexus enables improved detection of in vivo transcription factor binding footprints. *Nature Biotechnology* **33**, 395 (2015). URL `http://dx.doi.org/10.1038/nbt.3121`.

[83] Bailey, T. *et al.* Practical Guidelines for the Comprehensive Analysis of ChIP-seq Data. *PLOS Computational Biology* **9**, e1003326– (2013). URL `https://doi.org/10.1371/journal.pcbi.1003326`.

[84] de Koning, A. P. J., Gu, W., Castoe, T. A., Batzer, M. A. & Pollock, D. D. Repetitive Elements May Comprise Over Two-Thirds of the Human Genome. *PLOS Genetics* **7**, e1002384– (2011). URL `https://doi.org/10.1371/journal.pgen.1002384`.

[85] Langmead, B., Trapnell, C., Pop, M. & Salzberg, S. L. Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome Biology* **10**, R25 (2009). URL `https://doi.org/10.1186/gb-2009-10-3-r25`.

[86] Li, H. & Durbin, R. Fast and accurate short read alignment with Burrows–Wheeler transform. *Bioinformatics* **25**, 1754–1760 (2009). URL `http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2705234/`.

[87] Dobin, A. *et al.* STAR: ultrafast universal RNA-seq aligner. *Bioinformatics* **29**, 15–21 (2013). URL `http://dx.doi.org/10.1093/bioinformatics/bts635`.

[88] Jothi, R., Cuddapah, S., Barski, A., Cui, K. & Zhao, K. Genome-wide identification of in vivo protein–DNA binding sites from ChIP-Seq data. *Nucleic Acids Research* **36**, 5221–5231 (2008). URL `http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2532738/`.

[89] Wu, S., Wang, J., Zhao, W., Pounds, S. & Cheng, C. ChIP-PaM: an algorithm to identify protein-DNA interaction using ChIP-Seq data. *Theoretical Biology & Medical Modelling* **7**, 18 (2010). URL `http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2893127/`.

[90] 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**, 2344–2349 (2008). URL `http://dx.doi.org/10.1093/bioinformatics/btn402`.

[91] Nix, D. A., Courdy, S. J. & Boucher, K. M. Empirical methods for controlling false positives and estimating confidence in ChIP-Seq peaks. *BMC Bioinformatics* **9**, 523 (2008). URL `https://doi.org/10.1186/1471-2105-9-523`.

[92] Feng, X., Grossman, R. & Stein, L. PeakRanger: A cloud-enabled peak caller for ChIP-seq data. *BMC Bioinformatics* **12**, 139 (2011). URL `https://doi.org/10.1186/1471-2105-12-139`.

[93] Tuteja, G., White, P., Schug, J. & Kaestner, K. H. Extracting transcription factor targets from ChIP-Seq data. *Nucleic Acids Research* **37**, e113–e113 (2009). URL `http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2761252/`.

[94] Schweikert, G., Cseke, B., Clouaire, T., Bird, A. & Sanguinetti, G. MMDiff: quantitative testing for shape changes in ChIP-Seq data sets. *BMC Genomics* **14**, 826 (2013). URL `http://www.ncbi.nlm.nih.gov/pmc/articles/PMC4008153/`.

[95] Kharchenko, P. V., Tolstorukov, M. Y. & Park, P. J. Design and analysis of ChIP-seq experiments for DNA-binding proteins. *Nature biotechnology* **26**, 1351–1359 (2008). URL `http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2597701/`.

[96] Fejes, A. P. *et al.* FindPeaks 3.1: a tool for identifying areas of enrichment from massively parallel short-read sequencing technology. *Bioinformatics* **24**, 1729–1730 (2008). URL `http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2638869/`.

[97] Rashid, N. U., Giresi, P. G., Ibrahim, J. G., Sun, W. & Lieb, J. D. ZINBA integrates local covariates with DNA-seq data to identify broad and narrow regions of enrichment, even within amplified genomic regions. *Genome Biology* **12**, R67 (2011). URL `https://doi.org/10.1186/gb-2011-12-7-r67`.

[98] 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 Research* **42**, e95–e95 (2014). URL `http://www.ncbi.nlm.nih.gov/pmc/articles/PMC4066778/`.

[99] Zhang, Y., Lin, Y.-H., Johnson, T. D., Rozek, L. S. & Sartor, M. A. PePr: a peak-calling prioritization pipeline to identify consistent or differential peaks from replicated ChIP-Seq data. *Bioinformatics* **30**, 2568–2575 (2014). URL `http://www.ncbi.nlm.nih.gov/pmc/articles/PMC4155259/`.

[100] Rozowsky, J. *et al.* PeakSeq: Systematic Scoring of ChIP-Seq Experiments Relative to Controls. *Nature biotechnology* **27**, 66–75 (2009). URL `http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2924752/`.

[101] Shen, L. *et al.* diffReps: Detecting Differential Chromatin Modification Sites from ChIP-seq Data with Biological Replicates. *PLOS ONE* **8**, e65598– (2013). URL `https://doi.org/10.1371/journal.pone.0065598`.

[102] Wang, C., Xu, J., Zhang, D., Wilson, Z. A. & Zhang, D. An effective approach for identification of in vivo protein-DNA binding sites from paired-end ChIP-Seq data. *BMC Bioinformatics* **11**, 81 (2010). URL `http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2831849/`.

[103] Ji, H. *et al.* An integrated system CisGenome for analyzing ChIP-chip and ChIP-seq data. *Nature biotechnology* **26**, 1293–1300 (2008). URL `http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2596672/`.

[104] Zhang, Y. *et al.* Model-based Analysis of ChIP-Seq (MACS). *Genome Biology* **9**, R137 (2008). URL `https://doi.org/10.1186/gb-2008-9-9-r137`.

[105] Robinson, M. D. & Oshlack, A. A scaling normalization method for differential expression analysis of RNA-seq data. *Genome Biology* **11**, R25 (2010). URL `https://doi.org/10.1186/gb-2010-11-3-r25`.

[106] Xu, H. *et al.* A signal–noise model for significance analysis of ChIP-seq with negative control. *Bioinformatics* **26**, 1199–1204 (2010). URL `http://dx.doi.org/10.1093/bioinformatics/btq128`.

[107] Liang, K. & Keleş, S. Normalization of ChIP-seq data with control. *BMC Bioinformatics* **13**, 199 (2012). URL `https://doi.org/10.1186/1471-2105-13-199`.

[108] Shao, Z., Zhang, Y., Yuan, G.-C., Orkin, S. H. & Waxman, D. J. MAnorm: a robust model for quantitative comparison of ChIP-Seq data sets. *Genome Biology* **13**, R16 (2012). URL `https://doi.org/10.1186/gb-2012-13-3-r16`.

[109] Allhoff, M., Seré, K., F. Pires, J., Zenke, M. & G. Costa, I. Differential peak calling of ChIP-seq signals with replicates with THOR. *Nucleic Acids Research* **44**, e153–e153 (2016). URL `http://www.ncbi.nlm.nih.gov/pmc/articles/PMC5175345/`.

[110] Taslim, C. *et al.* Comparative study on ChIP-seq data: normalization and binding pattern characterization. *Bioinformatics* **25**, 2334–2340 (2009). URL `http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2800347/`.

[111] Mendoza-Parra, M. A., Sankar, M., Walia, M. & Gronemeyer, H. POLYPHEMUS: R package for comparative analysis of RNA polymerase II ChIP-seq profiles by non-linear normalization. *Nucleic Acids Research* **40**, e30–e30 (2012). URL `http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3287170/`.

[112] Hon, G., Ren, B. & Wang, W. ChromaSig: A Probabilistic Approach to Finding Common Chromatin Signatures in the Human Genome. *PLOS Computational Biology* **4**, e1000201– (2008). URL `https://doi.org/10.1371/journal.pcbi.1000201`.

[113] Muiño, J. M., Kaufmann, K., van Ham, R. C. H. J., Angenent, G. C. & Krajewski, P. ChIP-seq Analysis in R (CSAR): An R package for the statistical detection of protein-bound genomic regions. *Plant Methods* **7**, 11 (2011). URL `http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3114017/`.

[114] Cheung, M.-S., Down, T. A., Latorre, I. & Ahringer, J. Systematic bias in high-throughput sequencing data and its correction by BEADS. *Nucleic Acids Research* **39**, e103–e103 (2011). URL `http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3159482/`.

[115] Qin, Z. S. *et al.* HPeak: an HMM-based algorithm for defining read-enriched regions in ChIP-Seq data. *BMC Bioinformatics* **11**, 369 (2010). URL `https://doi.org/10.1186/1471-2105-11-369`.

[116] Song, Q. & Smith, A. D. Identifying dispersed epigenomic domains from ChIP-Seq data. *Bioinformatics* **27**, 870–871 (2011). URL `http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3051331/`.

[117] Spyrou, C., Stark, R., Lynch, A. G. & Tavaré, S. BayesPeak: Bayesian analysis of ChIP-seq data. *BMC Bioinformatics* **10**, 299 (2009). URL `http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2760534/`.

[118] Valouev, A. *et al.* Genome-Wide Analysis of Transcription Factor Binding Sites Based on ChIP-Seq Data. *Nature methods* **5**, 829–834 (2008). URL `http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2917543/`.

[119] Boyle, A. P., Guinney, J., Crawford, G. E. & Furey, T. S. F-Seq: a feature density estimator for high-throughput sequence tags. *Bioinformatics* **24**, 2537–2538 (2008). URL `http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2732284/`.

[120] Lun, D. S., Sherrid, A., Weiner, B., Sherman, D. R. & Galagan, J. E. A blind deconvolution approach to high-resolution mapping of transcription factor binding sites from ChIP-seq data. *Genome Biology* **10**, R142–R142 (2009). URL `http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2812949/`.

[121] Thornton, J. L. *et al.* Context dependency of Set1/ COMPASS-mediated histone H3 Lys4 trimethylation. *Genes & Development* **28**, 115–120 (2014).

[122] Feng, J. *et al.* Chronic cocaine-regulated epigenomic changes in mouse nucleus accumbens. *Genome Biology* **15**, R65–R65 (2014). URL `http://www.ncbi.nlm.nih.gov/pmc/articles/PMC4073058/`.

[123] Koues, O. I. *et al.* Enhancer Sequence Variants and Transcription Factor Deregulation Synergize to Construct Pathogenic Regulatory Circuits in B Cell Lymphoma. *Immunity* **42**, 186–198 (2015). URL `http://www.ncbi.nlm.nih.gov/pmc/articles/PMC4302272/`.

[124] Anders, S. & Huber, W. Differential expression analysis for sequence count data. *Genome Biology* **11**, R106 (2010). URL https://doi.org/10.1186/gb-2010-11-10-r106.

[125] Robinson, M. D., McCarthy, D. J. & Smyth, G. K. edgeR: a Bioconductor package for differential expression analysis of digital gene expression data. *Bioinformatics* **26**, 139–140 (2010). URL http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2796818/.

[126] Guo, Y., Mahony, S. & Gifford, D. K. High Resolution Genome Wide Binding Event Finding and Motif Discovery Reveals Transcription Factor Spatial Binding Constraints. *PLOS Computational Biology* **8**, e1002638– (2012). URL https://doi.org/10.1371/journal.pcbi.1002638.

[127] Bailey, T. L. *et al.* MEME Suite: tools for motif discovery and searching. *Nucleic Acids Research* **37**, W202–W208 (2009). URL http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2703892/.

[128] Georgiev, S. *et al.* Evidence-ranked motif identification. *Genome Biology* **11**, R19 (2010). URL https://doi.org/10.1186/gb-2010-11-2-r19.

[129] Alipanahi, B., Delong, A., Weirauch, M. T. & Frey, B. J. Predicting the sequence specificities of DNA- and RNA-binding proteins by deep learning. *Nature Biotechnology* **33**, 831 (2015). URL http://dx.doi.org/10.1038/nbt.3300.

[130] Avsec, Ž., Barekatain, M., Cheng, J. & Gagneur, J. Modeling positional effects of regulatory sequences with spline transformations increases prediction accuracy of deep neural networks. *Bioinformatics* btx727–btx727 (2017). URL http://dx.doi.org/10.1093/bioinformatics/btx727.

[131] Zhou, J. & Troyanskaya, O. G. Predicting effects of noncoding variants with deep learning–based sequence model. *Nature Methods* **12**, 931 (2015). URL http://dx.doi.org/10.1038/nmeth.3547.

[132] Laajala, T. D. *et al.* A practical comparison of methods for detecting transcription factor binding sites in ChIP-seq experiments. *BMC Genomics* **10**, 618 (2009). URL http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2804666/.

[133] Wilbanks, E. G. & Facciotti, M. T. Evaluation of Algorithm Performance in ChIP-Seq Peak Detection. *PLOS ONE* **5**, e11471– (2010). URL https://doi.org/10.1371/journal.pone.0011471.

[134] Rye, M. B., Sætrom, P. & Drabløs, F. A manually curated ChIP-seq benchmark demonstrates room for improvement in current peak-finder programs. *Nucleic Acids Research* **39** (2011).

[135] Micsinai, M. *et al.* Picking ChIP-seq peak detectors for analyzing chromatin modification experiments. *Nucleic Acids Research* **40**, e70–e70 (2012). URL http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3351193/.

[136] Koohy, H., Down, T. A., Spivakov, M. & Hubbard, T. A Comparison of Peak Callers Used for DNase-Seq Data. *PLOS ONE* **9**, e96303– (2014). URL https://doi.org/10.1371/journal.pone.0096303.

[137] Szalkowski, A. M. & Schmid, C. D. Rapid innovation in ChIP-seq peak-calling algorithms is outdistancing benchmarking efforts. *Briefings in Bioinformatics* **12**, 626–633 (2011). URL http://dx.doi.org/10.1093/bib/bbq068.

[138] Thomas, R., Thomas, S., Holloway, A. K. & Pollard, K. S. Features that define the best ChIP-seq peak calling algorithms. *Briefings in Bioinformatics* **18**, 441–450 (2017). URL http://dx.doi.org/10.1093/bib/bbw035.

[139] Steinhauser, S., Kurzawa, N., Eils, R. & Herrmann, C. A comprehensive comparison of tools for differential ChIP-seq analysis. *Briefings in Bioinformatics* **17**, 953–966 (2016). URL http://dx.doi.org/10.1093/bib/bbv110.

[140] Nelder, J. A. & Wedderburn, R. W. M. Generalized Linear Models. *Journal of the Royal Statistical Society. Series A (General) Journal of the Royal Statistical Society. Series A (General J. R. Statist. Soc. A* **13517213**, 370–384 (1972). URL http://www.jstor.org/stable/2344614.

[141] Love, M. I., Huber, W. & Anders, S. Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2. *Genome Biology* **15**, 550 (2014). URL https://doi.org/10.1186/s13059-014-0550-8.

[142] Xing, H., Mo, Y., Liao, W. & Zhang, M. Q. Genome-Wide Localization of Protein-DNA Binding and Histone Modification by a Bayesian Change-Point Method with ChIP-seq Data. *PLOS Computational Biology* **8**, e1002613– (2012). URL https://doi.org/10.1371/journal.pcbi.1002613.

[143] Zambelli, F., Pesole, G. & Pavesi, G. Motif discovery and transcription factor binding sites before and after the next-generation sequencing era. *Briefings in Bioinformatics* **14**, 225–237 (2013). URL http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3603212/.

[144] Hoang, S. A., Xu, X. & Bekiranov, S. Quantification of histone modification ChIP-seq enrichment for data mining and machine learning applications. *BMC Research Notes* **4**, 288 (2011). URL http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3170335/.

[145] Ibrahim, M. M., Lacadie, S. A. & Ohler, U. JAMM: a peak finder for joint analysis of NGS replicates. *Bioinformatics* **31**, 48–55 (2015). URL http://dx.doi.org/10.1093/bioinformatics/btu568.

[146] Shimazaki, H. & Shinomoto, S. A Method for Selecting the Bin Size of a Time Histogram. *Neural Computation* **19**, 1503–1527 (2007). URL `https://doi.org/10.1162/neco.2007.19.6.1503`.

[147] Chitpin, J. G., Awdeh, A. & Perkins, T. J. RECAP reveals the true statistical significance of ChIP-seq peak calls. *bioRxiv* (2018).

[148] Li, Q., Brown, J. B., Huang, H. & Bickel, P. J. Measuring reproducibility of high-throughput experiments. *Ann. Appl. Stat.* **5**, 1752–1779 (2011). URL `https://projecteuclid.org:443/euclid.aoas/1318514284`.

[149] Hastie, T. & Tibshirani, R. Generalized Additive Models. *Statistical Science* **1**, 297 – 318 (1986).

[150] Hastie, T. & Tibshirani, R. *Generalized Additive Models* (Chapman and Hall/CRC, New York, 1990).

[151] Wood, S. *Generalized Additive Models: An Introduction with R* (Chapman and Hall/CRC, New York, 2017), 2 edn.

[152] McCullagh, P. & Nelder, J. *Generalized Linear Models* (Chapman and Hall/CRC, London, 1989), 2nd edn.

[153] Thurston, S. W., Wand, M. P. & Wiencke ', J. K. Negative Binomial Additive Models. *Biometrics* **56**, 139–144 (2000).

[154] De Boor, C. *A practical guide to splines*, vol. 27 (Springer-Verlag New York, 1978).

[155] Eilers, P. H. C. & Marx, B. D. Flexible smoothing with B -splines and penalties (1996).

[156] Davis, T. *Direct Methods for Sparse Linear Systems* (Society for Industrial and Applied Mathematics, 2006). URL `http://epubs.siam.org/doi/abs/10.1137/1.9780898718881`.

[157] Zaharia, M., Chowdhury, M., Franklin, M. J., Shenker, S. & Stoica, I. Spark: cluster computing with working sets. In *Proceedings of the 2nd USENIX conference on Hot topics in cloud computing*, vol. 10, 10 (2010).

[158] Nelder, J. A. & Mead, R. A Simplex Method for Function Minimization. *The Computer Journal* **7**, 308–313 (1965). URL `http://dx.doi.org/10.1093/comjnl/7.4.308`.

[159] Meyer, C. A. & Shirley Liu, X. Identifying and mitigating bias in next-generation sequencing methods for chromatin biology. *Nature Publishing Group* **15** (2014).

[160] Bourgon, R., Gentleman, R. & Huber, W. Independent filtering increases detection power for high-throughput experiments. *Proceedings of the National Academy of Sciences* **107**, 9546–9551 (2010).

[161] Hochberg, Y. A sharper Bonferroni procedure for multiple tests of significance. *Biometrika* **75**, 800–802 (1988).

[162] Benjamini, Y. & Hochberg, Y. Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing. *Journal of the Royal Statistical Society. Series B (Methodological)* **57**, 289–300 (1995). URL `http://www.jstor.org/stable/2346101http://www.jstor.org/http://www.jstor.org/action/showPublisher?publisherCode=black`.

[163] Marra, G. & Wood, S. N. Coverage properties of confidence intervals for generalized additive model components. *Scandinavian Journal of Statistics* **39**, 53–74 (2012).

[164] Wei, Z., Sun, W., Wang, K. & Hakonarson, H. Multiple testing in genome-wide association studies via hidden Markov models. *Bioinformatics (Oxford, England)* **25**, 2802–2808 (2009). URL `http://bioinformatics.oxfordjournals.org/cgi/doi/10.1093/bioinformatics/btp476`.

[165] Xu, Z. *et al.* Bidirectional promoters generate pervasive transcription in yeast. *Nature* (2009). URL `http://www.nature.com/nature/journal/vaop/ncurrent/full/nature07728.html`.

[166] Huber *et al.* Orchestrating high-throughput genomic analysis with Bioconductor. *Nature Methods* **12**, 115–121 (2015). URL `http://www.nature.com/nmeth/journal/v12/n2/full/nmeth.3252.html`.

[167] Wood, S. N. Fast stable restricted maximum likelihood and marginal likelihood estimation of semiparametric generalized linear models. *Journal of the Royal Statistical Society. Series B: Statistical Methodology* (2011).

[168] Li, X. S. An Overview of SuperLU: Algorithms, Implementation, and User Interface **31**, 302–325 (2005).

[169] Campbell, Y. E. & Davis, T. A. Computing the Sparse Inverse Subset : An inverse multifrontal approach. Tech. Rep., Computer and Information Sciences Deparment, University of Florida, Gainesville, FL (1995).

[170] Duff, I. S. & Reid, J. K. The Multifrontal Solution of Indefinite Sparse Symmetric Linear. *ACM Trans. Math. Softw.* **9**, 302–325 (1983). URL `http://doi.acm.org/10.1145/356044.356047`.

[171] Liu, J. The Role of Elimination Trees in Sparse Factorization. *SIAM Journal on Matrix Analysis and Applications* **11**, 134–172 (1990). URL `https://doi.org/10.1137/0611010`.

[172] Rue, H. & Held, L. *Gaussian Markov Random Fields: Theory and Applications* (Chapman and Hall/CRC, Boca Rota, 2005).

[173] Zammit-Mangion, A. sparseinv: Computation of the Sparse Inverse Subset (2018).

[174] Davis, T. A. SuiteSparse: A suite of sparse matrix software. URL `http://faculty.cse.tamu.edu/davis/suitesparse.html`.

[175] The HDF Group. Hierarchical data format version 5. URL `http://www.hdfgroup.org/HDF5`.

[176] Pagès, H. HDF5Array: HDF5 back end for DelayedArray objects (2017).

[177] Fischer, B., Pau, G. & Smith, M. rhdf5: HDF5 interface to R (2017).

[178] Bates, D. & Maechler, M. Matrix: Sparse and Dense Matrix Classes and Methods (2017). URL `https://cran.r-project.org/package=Matrix`.

[179] Morgan, M., Obenchain, V., Lang, M. & Thompson, R. BiocParallel: Bioconductor facilities for parallel evaluation (2017). URL `https://github.com/Bioconductor/BiocParallel`.

[180] Lee, C. Y. Y. & Wand, M. P. Streamlined mean field variational Bayes for longitudinal and multilevel data analysis. *Biometrical Journal* **58**, 868 – 895 (2016).

[181] Wood, S. N. Comment. *Journal of the American Statistical Association* **112**, 164 – 166 (2017).

[182] Basehoar, A. D., Zanton, S. J. & Pugh, B. F. Identification and distinct regulation of yeast TATA box-containing genes. *Cell* **116**, 699–709 (2004).

[183] Wedderburn, R. W. M. Quasi-likelihood functions, generalized linear models, and the Gauss—Newton method. *Biometrika* **61**, 439–447 (1974).

[184] Smallwood, S. A. *et al.* Single-cell genome-wide bisulfite sequencing for assessing epigenetic heterogeneity. *Nature methods* **11**, 817–820 (2014).

[185] Heinis, T. Data analysis: approximation aids handling of big data. *Nature* **515**, 198 (2014). URL `http://dx.doi.org/10.1038/515198d`.

[186] Schulz, D. *et al.* Transcriptome Surveillance by Selective Termination of Noncoding RNA Synthesis. *Cell* **155**, 1075–1087 (2013).

[187] Goecks, J., Nekrutenko, A. & Taylor, J. Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. *Genome Biology* **11**, R86 (2010). URL `http://genomebiology.com/2010/11/8/R86`.

[188] Langmead, B., Trapnell, C., Pop, M. & Salzberg, S. L. Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome Biology* **10**, R25 (2009). URL `http://genomebiology.com/2009/10/3/R25`.

[189] Grant, C. E., Bailey, T. L. & Noble, W. S. FIMO: scanning for occurrences of a given motif. *Bioinformatics (Oxford, England)* **27**, 1017–1018 (2011). URL `http://bioinformatics.oxfordjournals.org/content/27/7/1017`.

[190] Mathelier, A. *et al.* JASPAR 2014: an extensively expanded and updated open-access database of transcription factor binding profiles. *Nucleic acids research* **42**, D142–7 (2014). URL `http://nar.oxfordjournals.org/content/early/2013/11/04/nar.gkt997.full`.