# Efficient Estimation Algorithms for Large and Complex Data Sets

**Alexander Engelhardt**


München 2017

# Efficient Estimation Algorithms for Large and Complex Data Sets

**Alexander Engelhardt**

Dissertation
an der Fakultät für Mathematik, Informatik und Statistik
der Ludwig–Maximilians–Universität
München

vorgelegt von
Alexander Engelhardt
am 18. April 2017
in München

Erstgutachter: Prof. Dr. Ulrich Mansmann
Zweitgutachterin: Prof. Dr. Heike Bickeböller
Tag der Disputation: 28. Juli 2017

# Summary

The recent world-wide surge in available data allows the investigation of many new and sophisticated questions that were inconceivable just a few years ago. However, two types of data sets often complicate the subsequent analysis: Data that is simple in structure but large in size, and data that is small in size but complex in structure.

These two kinds of problems also apply to biological data. For example, data sets acquired from family studies, where the data can be visualized as pedigrees, are small in size but, because of the dependencies within families, they are complex in structure. By comparison, next-generation sequencing data, such as data from chromatin immunoprecipitation followed by deep sequencing (ChIP-Seq), is simple in structure but large in size. Even though the available computational power is increasing steadily, it often cannot keep up with the massive amounts of new data that are being acquired. In these situations, ordinary methods are no longer applicable or scale badly with increasing sample size. The challenge in today's environment is then to adapt common algorithms for modern data sets.

This dissertation considers the challenge of performing inference on modern data sets, and approaches the problem in two parts: first using a problem in the field of genetics, and then using one from molecular biology.

In the first part, we focus on data of a complex nature. Specifically, we analyze data from a family study on colorectal cancer (CRC). To model familial clusters of increased cancer risk, we assume inheritable but latent variables for a risk factor that increases the hazard rate for the occurrence of CRC. During parameter estimation, the inheritability of this latent variable necessitates a marginalization of the likelihood that is costly in time for large families. We first approached this problem by implementing computational accelerations that reduced the time for an optimization by the Nelder-Mead method to about 10% of a naive implementation. In a next step, we developed an expectation-maximization (EM) algorithm that works on data obtained from pedigrees. To achieve this, we used factor graphs to factorize the likelihood into a product of "local" functions, which enabled us to apply the sum-product algorithm in the E-step, reducing the computational complexity from exponential to linear. Our algorithm thus enables parameter estimation for family studies in a feasible amount of time.

In the second part, we turn to ChIP-Seq data. Previously, practitioners were required to assemble a set of tools based on different statistical assumptions and dedicated to specific applications such as calling protein occupancy peaks or testing for differential occupancies between experimental conditions. In order to remove these restrictions and create a unified framework for ChIP-Seq analysis, we developed GenoGAM (Genome-wide Generalized Additive Model), which extends generalized additive models to efficiently work on data spread over a long $x$ axis by reducing the scaling from cubic to linear and by employing a data parallelism strategy. Our software makes the well-established and flexible GAM framework available for a number of genomic applications. Furthermore, the statistical framework allows for significance testing for differential occupancy.

In conclusion, I show how developing algorithms of lower complexity can open the door for analyses that were previously intractable. On this basis, it is recommended to focus subsequent research efforts on lowering the complexity of existing algorithms and design new, lower-complexity algorithms.

# Zusammenfassung

Der jüngste weltweite Anstieg an verfügbaren Daten ermöglicht die Untersuchung vieler neuer und anspruchsvoller Fragen, die vor wenigen Jahren noch undenkbar waren. Allerdings erschweren zwei Arten von Datensätzen oft die nachfolgende Analyse: Daten, die einfach in der Struktur, aber groß sind, und Daten, die klein aber komplex in der Struktur sind.

Diese beiden Arten von Problemen treten auch bei biologische Daten auf. Zum Beispiel sind Datensätze aus Familienstudien, in denen die Daten als Stammbäume visualisiert werden können, klein, aber – aufgrund der Abhängigkeiten innerhalb der Familien, komplex in der Struktur. Im Vergleich dazu sind Next-Generation-Sequencing-Daten, wie z.B. ChIP-Seq-Daten, strukturell simpel, aber sehr groß. Obwohl die verfügbare Rechenleistung stetig zunimmt, kann sie oft nicht mit den massiven Mengen an neuen Daten Schritt halten. In diesen Situationen sind herkömmliche Methoden nicht mehr anwendbar oder skalieren schlecht mit zunehmender Stichprobengröße. Die Herausforderung in der heutigen Umgebung besteht darin, verbreitete Algorithmen an moderne Datensätze anzupassen.

Diese Dissertation betrachtet die Herausforderung, Inferenz auf modernen Datensätze zu betreiben und nähert sich dem Problem in zwei Teilen: zunächst mit einem Problem auf dem Gebiet der Genetik und dann mit einem aus der Molekularbiologie.

Im ersten Teil konzentrieren wir uns auf komplexe Daten. Wir analysieren Daten aus einer Familienstudie über Darmkrebs (CRC). Um familiäre Cluster eines erhöhten Krebsrisikos zu modellieren, nehmen wir vererbbare aber latente Variablen für einen Risikofaktor an, der die Hazardrate für das Auftreten von CRC erhöht. In der Parameterschätzung erfordert die Vererbbarkeit dieser latenten Variablen eine Marginalisierung der Likelihood, die zeitaufwändig für große Familien ist. Wir gingen dieses Problem zunächst an, indem wir Rechenbeschleunigungen implementierten, die die Zeit für eine Optimierung durch die Nelder-Mead-Methode auf etwa 10% einer naiven Implementierung reduzierten. In einem nächsten Schritt entwickelten wir einen Expectation-Maximization (EM) Algorithmus, der auf Stammbaumdaten arbeitet. Um dies zu erreichen, verwendeten wir Faktorgraphen, um die Likelihood in ein Produkt von "lokalen" Funktionen zu zerlegen, was es uns ermöglichte, den Sum-Product-Algorithmus im E-Schritt anzuwenden, wodurch die rechnerische Komplexität von exponentiell auf linear reduziert wurde. Unser Algorithmus ermöglicht somit eine Parameterschätzung für Familienstudien in einem durchführbaren Zeitraum.

Im zweiten Teil wenden wir uns ChIP-Seq-Daten zu. Bisher waren Anwender gezwungen, mehrere verschiedene Softwaretools zu verwenden, die auf verschiedenen statistischen Annahmen basieren, und jeweils für spezifische Anwendungen geschaffen wurde, wie z.B. das peak-calling, oder das Finden von Regionen mit differenzieller Bindung. Um diese Einschränkungen zu beseitigen und einen einheitlichen Rahmen für die ChIP-Seq-Analyse zu schaffen, entwickelten wir GenoGAM (genome-wide generalized additive models), das klassische generalisierte additive Modelle erweitert, um effizient mit Daten auf einer langen $x$-Achse arbeiten zu können. Dies erreichten wir, indem die Skalierung von kubisch zu linear verringert wurde, und durch das Verwenden eines parallelisierten Algorithmus. Unsere Software stellt das bewährte und flexible GAM-Framework für eine Vielzahl von genomischen Anwendungen zur Verfügung. Darüber hinaus ermöglicht das statistische Framework Signifikanztests für differentielle Bindung.

Zusammenfassend zeige ich, wie man durch die Entwicklung von Algorithmen mit niedriger Komplexität die Tür für Analysen öffnen kann, die bisher nicht machbar waren. Auf dieser Grundlage empfiehlt es sich, nachfolgende Forschung auf die Senkung der Komplexität bestehender Algorithmen und die Konstruktion neuer Algorithmen mit geringerer Komplexität zu konzentrieren.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

The Digital Revolution that occurred between the 1950s and 1970s marked the beginning of the Information Age [101]. Most people now use information technology on a daily basis, knowingly or unknowingly. Naturally, this development has resulted in vast amounts of data being generated and collected. According to IBM, 90% of all available data in the world has been created in the last two years [50].

Modern data sets have grown both more complex and greatly increased in size. Complex data sets include pedigrees, which have been analyzed in the field of genetics for decades [65]. More recent developments allow the analysis of larger pedigrees and features such as quantitative traits [54]. An adequate analysis of these complex data sets can prepare and create new breakthroughs in science, government administration, and industry. However, while it is possible to analyze this kind of data with standard methods, doing so disregards a great deal of knowledge that could be extracted with more advanced methods [35].

Data sets that are simple in structure but large in size can in theory still be analyzed with classical statistical methods, but would require an infeasibly long runtime of the analysis. For example, the advent of next-generation sequencing (NGS) technology in molecular biology allowed low-priced sequencing of whole genomes and thus opened the door for studies of the relationship between DNA and complex phenotypes such as cancer development [41]. The generated data in this case is rather simple in structure, but, in the case of the human genome, for example, spans over an $x$-axis of 3 billion discrete points. In these cases, it is necessary to revisit established methods and enhance their computational efficiency so that they can efficiently deal with large data sets.

We see from these two examples that the complexity and size of modern data sets present a major obstacle to their statistical analysis. In the last 20 years, there has been tremendous progress in the development of new analysis methods, both in the areas of statistics and computer science. A currently popular approach is to parallelize algorithms. As we will see in Chapter 2, parallelization does deliver time savings, but it does not reduce the complexity of algorithms. Instead, in this thesis we focus on developing efficient analysis

methods by reducing the complexity of existing algorithms.

To that end, I explore two aspects of working with modern data sets: First, I focus on complex, partially dependent data sets in the form of pedigrees, and develop an expectation-maximization (EM) algorithm to efficiently estimate parameters on this data. Second, I investigate large genomic data that is simple in structure, but because of its enormous size, classical statistical models can not be directly applied to it.

The remainder of this thesis is structured as follows:

Chapter 2 explains and discusses some topics that are necessary to understand the rest of the thesis. It discusses the time complexity of algorithms, the big O notation, and common approaches used to speed up programs, including parallelization. Chapter 3 presents an EM algorithm for pedigree data in which a straightforward Maximum-Likelihood estimation would suffer from exponential runtime increase respective to the family sizes. In Chapter 4, I develop a parallelized algorithm for applying Generalized Additive Models (GAMs) on genomic data by using a split-apply-combine approach [104] to fit separate models on "chunks" of data and subsequently combine them into one big model. Chapter 5 is a short conclusion of the thesis that summarizes its contributions. Finally, the appendix contains some supplementary material on the methods described in Chapters 3 and 4.

Parts of this thesis have been submitted for publication as articles in peer-reviewed journals. Specifically, the articles and the respective authors' contributions are as follows:

- A. Engelhardt, A. Rieger, A. Tresch, U. Mansmann. **Efficient Maximum Likelihood Estimation for Pedigree Data with the Sum-Product Algorithm.** *Human Heredity*, 2017 (in press). DOI: `https://doi.org/10.1159/000475465`

  Alexander Engelhardt developed the software in R, carried out all analyses, and wrote the manuscript. Anna Rieger provided the processed data and gave advice on data processing. Achim Tresch gave advice on the sum-product algorithm. Ulrich Mansmann conceived and supervised this study.

- G. Stricker*, A. Engelhardt*, D. Schulz, M. Schmid, A. Tresch, J. Gagneur. **GenoGAM: Genome-Wide Generalized Additive Models for ChIP-Seq Analysis.** *Bioinformatics*, 2017. DOI: `https://doi.org/10.1093/bioinformatics/btx150`

  * These authors contributed equally to this work.

  Alexander Engelhardt and Georg Stricker developed the software in R and carried out all analyses. The manuscript was mainly written by Alexander Engelhardt, Georg Stricker and Julien Gagneur. Daniel Schulz carried out the ChIP-Seq experiments for TFIIB on yeast. Matthias Schmid gave advice on statistics. Julien Gagneur and Achim Tresch initiated and supervised this project.

This dissertation is of course much more elaborate than the sections submitted to the journals, where brevity was important. I am pleased to have more room available here

to discuss the background more deeply and show a few more examples, annotations, and figures.

# Chapter 2

# Methodological Foundations

## 2.1 Chapter summary

Because the focus of this thesis is the reduction of the runtime of estimation algorithms, this chapter lays the foundation for the remainder of the thesis by explaining the time complexity of algorithms (Section 2.2) followed by an explanation of the big O notation (Section 2.3), which is the standard way of quantifying the runtime of an algorithm. Next, we review common approaches to improve the runtime. In Section 2.4 we turn to parallelization, currently the most popular method used to improve algorithm runtime, and discuss its advantages and drawbacks. Finally, in Section 2.5, I argue that in time-critical cases, instead of just parallelizing an existing algorithm, it is often more fruitful to lower the time complexity of the algorithm itself. This conclusion then directly leads into the two projects that comprise the main part of this thesis.

## 2.2 Time complexity of algorithms

Algorithms are developed to solve both simple and highly complex problems. For example, the question of whether a number represented as $n$ binary bits (such as `1100101`) is even or odd can be solved in a single step: Check whether the last digit is 1 or 0, and return "odd" or "even", respectively. This problem is independent of $n$ because for longer numbers, one still has to evaluate only the last binary digit.

The runtime of most algorithms is of course dependent on the input size. A simple example problem is determining the maximum of an integer vector of length $n$. A solution should take at least $n$ steps, because it has to look at each number once and determine whether it is greater than the currently stored maximum. If the input size $n$ increases by one, the runtime of this algorithm will also increase by one step.

An example of an even more difficult problem is the breaking of a rotary combination lock with $n$ numbers (Figure 2.1). For example, to find a combination of length $n = 4$ with 10 possible digits in each place, one needs $10^4$ tries at most. If the input size $n$ (i.e., the length of the combination) increases by one, the runtime increases by a *factor* of ten for each additional digit.

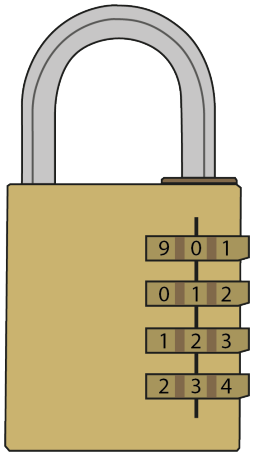



Figure 2.1: A combination lock with 4 digits is an example of an algorithm running in exponential time. Trying every combination would take $10^4$ steps (i.e., exponential time). However, if the lock provides feedback after each digit, it would take at most $4 \cdot 10$ steps (i.e., linear time) to find the combination.

Mathematically speaking, the time complexity of a problem, or equivalently, of an algorithm that solves this problem, is a function $T(n)$ that returns the maximum number of *steps* (CPU cycles, seconds, etc.) needed to solve an algorithm with an input of length $n$ [94, Chap. 7].

The exact time complexity is often too difficult to compute and is of little interest to the practitioner. Therefore, an asymptotic notation called the *big O notation* was established to classify time complexities into a few distinct classes, based on their asymptotic runtime behavior as the input size grows larger.

## 2.3 Big O notation

In most cases, it is enough to describe the *asymptotic* runtime of an algorithm as the input size grows larger. In these cases, it suffices to consider only the fastest-growing term in the expression for the runtime and disregard both the coefficient of that term as well as all lower-order terms [94].

For example, when the runtime of an algorithm with input size $n$ is $T(n) = 8n^4+2n^2+n+4$, we can shorten this expression and say that the runtime is $T(n) = \mathcal{O}(n^4)$.

More precisely, according to Sipser [94], we can asymptotically define the runtime $T(n)$ of an algorithm with input size $n$ as $T(n) = \mathcal{O}(g(n))$, if there exist positive integers $c$ and $n_0$ such that for every integer $n > n_0$

$$T(n) \leq c \cdot g(n).$$

Then, the function $g(n)$ is called an *asymptotic upper bound* for the runtime $T(n)$.

Coming back to the three examples from Section 2.2, the runtime for determining whether a number represented in $n$ binary digits is a constant number of steps (possibly including some sort of preparation). Since the big O notation ignores these constants, we can specify the runtime as $T(n) = \mathcal{O}(1)$, that is, it is some constant time independent of the input size $n$. Determining the maximum of a vector of length $n$ would be of complexity $T(n) = \mathcal{O}(n)$, and so breaking a $n$-digit combination lock with 10 possible digits per place is of complexity $T(n) = \mathcal{O}(10^n)$.

We can briefly denote the runtime of an algorithm of the complexity $\mathcal{O}(1)$ as *constant runtime*. Equivalently, we call $\mathcal{O}(n)$ *linear* runtime, $\mathcal{O}(n^d)$ polynomial runtime and $\mathcal{O}(d^n)$ *exponential* runtime. The distinction between $n^d$ and $d^n$ is crucial. If we set $d = 2$, the runtime of an $\mathcal{O}(n^2)$ algorithm grows quadratically, but an $\mathcal{O}(2^n)$ algorithm grows exponentially, meaning it increases much more rapidly as $n$ grows.

The beauty and convenience of the big O notation is that it disregards lower-order terms of the runtime. For example, if breaking a combination lock would take a constant 15 steps for some sort of setup, finding the correct combination takes $10^n$ steps, and writing down the solution takes $2n$ steps, two per digit, the actual runtime would be $T(n) = 15 + 10^n + 2n$, but in big O notation, it would still be $T(n) = \mathcal{O}(10^n)$.

## 2.4 Parallel computing

In 1965, Gordon Moore predicted that the number of transistors on a microprocessor, and thus their performance, will double roughly every two years for the foreseeable future [80]. This has come to be known as Moore's Law. Recently, this trend has begun to reach saturation. Starting in the 1990s, the period for a doubled transistor count increased to 2.5 years [86]. Physical constraints prohibit the indefinite growth of transistor count, and different strategies will have to be developed in the future [33].

To keep up with Moore's Law, chip manufacturers started producing multi-core processors, in which they put more than one processing core on a single chip. This way, multiple processing units can work on different problems at the same time. More interestingly, it is possible to can split up a single problem into independent subproblems and let each core solve only these parts of the problem. To make full use of this parallelism strategy, programmers have to develop their algorithms in a parallelizable manner, so that they can

be run on a multi-core machine [105]. Existing algorithms must also be rewritten to make use of a multi-core environment.

### 2.4.1 Advantages

The advantages of parallel computing are clear: By assigning, for example, 8 workers to a problem, the task can be finished up to 8 times faster. Furthermore, with the advent of *cluster computing*, it is now easy to parallelize algorithms across multiple machines, each with its own (possibly multi-core) processors. Thus, the limits on the possible number of cores in a parallel environment disappear.

A computing cluster is relatively easy to use but can be difficult to set up on the hardware side. Recently, the development of *cloud computing* has provided a solution to this problem. For example, Amazon offers the Elastic Compute Cloud (EC2), which allows users to rent virtual computers and run their programs on them [56]. As of 2017, the price for these computers is as low as a few cents per hour [6].

### 2.4.2 Limitations

Parallelization strategies suffer from two main drawbacks. First, the term *parallel slowdown* describes an overhead due to setting up the parallel environment, distributing the data, and communication between the processes ("housekeeping"). This parallel slowdown is described by Amdahl's Law [7], which implies that as the number of processors grows towards infinity, the execution time of a parallelized program does not shrink to zero but to some constant time.

Second, parallelizing a program does usually not reduce the order of the time complexity, that is, the order $\mathcal{O}(g(n))$ in big O notation. Instead, parallelizing an algorithm of complexity $\mathcal{O}(2^n)$ with 8 cores only divides its runtime by a constant of no more than 8, which means that the complexity is still $\mathcal{O}(2^n)$ (Figure 2.2).

## 2.5 Improving existing algorithms

In Section 2.4 we discussed that, while parallelization is a method to improve the speed of a program, it does not lower its computational complexity. Instead, it only improves the runtime by a constant factor.

It is often more beneficial to improve the implementation of an algorithm to lower its complexity. Unfortunately, this is a more challenging task than a parallelization, but in cases where it is possible, nonetheless it is often a recommendable approach.

To come back to the example of breaking a combination lock: A parallelized solution could be achieved if one had 10 replicates of the lock and 10 people trying different numbers at the same time. The runtime would be divided by 10 but still be $\mathcal{O}(10^n)$. On the other hand, if the lock had some manufacturing defect and gave mechanical feedback after each single digit that is correct, it would take at most $10 \cdot 4$ tries to break the lock (i.e., 10 per digit). The runtime would thus be reduced to linear (i.e., $\mathcal{O}(n)$), and would be much faster than any feasible parallelization, especially for larger $n$.

A real-world example for an improved implementation of a common algorithm can be taken from the field of sorting algorithms, where the task is to sort an array of $n$ numbers in ascending order. A naive algorithm is called *selection sort*, which iteratively walks through the whole vector, finds the currently smallest number in the unsorted part of the array, and exchanges it with the leftmost unsorted value. For an array of $n$ numbers, this takes $\frac{(n-1)(n-2)}{2}$ comparisons. The complexity of this algorithm is thus $\mathcal{O}(n^2)$ [58]. A faster algorithm called Quicksort [47] solves the same task with an average runtime of $\mathcal{O}(n \log n)$. This is achieved by using a *divide and conquer* approach, recursively dividing the array into a "low" and a "high" subgroup and sorting within these subgroups.

Figure 2.2 shows the schematic runtimes of four different algorithms. It is worth noting that the algorithms of lower complexity are not uniformly faster than a parallelized $\mathcal{O}(2^n)$ algorithm, but only asymptotically as $n$ grows large. On the log-scale, the curve for the runtime of the parallelized algorithm is just shifted down by a constant, whereas the curve for the improved algorithm of complexity $\mathcal{O}(n \log n)$ has a lower slope at all sufficiently large $n$.

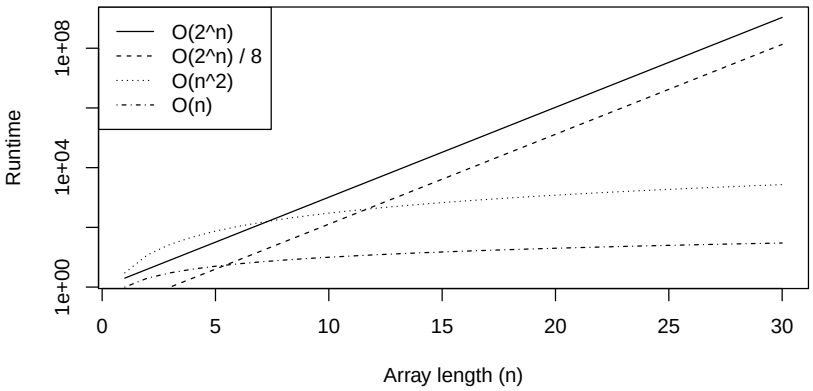



Figure 2.2: Schematic runtimes of algorithms of complexity $\mathcal{O}(2^n)$ (solid line), an 8-core parallelization of the same $\mathcal{O}(2^n)$ algorithm (dashed line), and an improved version of complexity $\mathcal{O}(n^2)$ (dotted line), and an even faster algorithm of complexity $\mathcal{O}(n)$ (dash-dotted line). The $y$-axis (log-scale) shows the runtime versus the input size $n$ on the $x$-axis.

To conclude this chapter, massively parallel computing with, say, 200 cores, can reduce the runtime of an algorithm that takes a week (10,080 minutes) to approximately 50 minutes

at best, which is mostly a question of convenience. On the other hand, reducing the complexity of an algorithm from, say, $\mathcal{O}(2^n)$ to $\mathcal{O}(n)$ can make problems solvable that were previously intractable.

# Chapter 3

# Efficient Maximum Likelihood Estimation for Pedigree Data with the Sum-Product Algorithm

## 3.1 Chapter summary

In this chapter, we analyze data sets consisting of pedigrees where the response is the age at onset of colorectal cancer (CRC). The occurrence of familial clusters of CRC suggests the existence of a latent, inheritable risk factor. We aim to compute the probability of a family possessing this risk factor, as well as the hazard rate increase for these risk factor carriers. Due to the inheritability of this risk factor, the estimation necessitates a costly marginalization of the likelihood.

We therefore develop an EM algorithm that works with dependent latent variables by applying factor graphs and the sum-product algorithm in the E-step, reducing the computational complexity from exponential to linear in the number of family members.

Our algorithm is as precise as a direct likelihood maximization in a simulation study and a real family study on CRC risk. For 250 simulated families of size 23 and 25, the runtime of our algorithm is faster by a factor of 13 and 39, respectively. On the largest family (23 members) in the real data, our algorithm is 4 times faster.

We introduce a flexible and runtime-efficient tool for statistical inference in biomedical event data that opens the door for advanced analyses of pedigree data.

All scripts developed for this chapter are available on GitHub at `http://github.com/AlexEngelhardt/sumproduct`.

## 3.2 Introduction

### 3.2.1 Familial CRC clusters and estimation of family risk

Colorectal cancer (CRC) is one of the most prevalent cancer diseases in Europe and the United States [57], with men having a younger average age at diagnosis [59].

For a small proportion of CRC cases, genetic predispositions are known [44]. Interestingly, an additional 15%–20% of CRC cases occur in familial clusters [76]. Within these clusters, family members show a higher risk of contracting CRC [18]. The cause for these clusters is unknown but assumed to be a risk factor which may be of genetic or environmental origin.

Since cancer develops earlier in these high-risk families, it is of interest to identify them in advance. Subsequently, health insurers can allow members of high-risk families to join screening programs at an earlier age. We will therefore develop an efficient risk calculator for CRC (i.e., a method clinicians can use to assess the familial risk for a specific family based on their CRC history).

### 3.2.2 Efficient Maximum Likelihood estimation (MLE) for partially dependent data

Here, we look at data consisting of a set of pedigrees, where each person has an inheritable latent variable, which is the risk factor that influences its response variable, or the age at CRC diagnosis. Assuming an inheritance model and a penetrance model, we aim to estimate two parameters: the a priori probability $p_1$ for a founder to carry the risk factor, and the penetrance $\alpha$, that is, the multiplicative increase of the hazard rate of an individual who carries the risk factor.

### 3.2.3 Existing work

A closely related subject is *complex segregation analysis* (CSA). CSA is a method used to evaluate whether pedigree data of affected and unaffected offspring agrees with a Mendelian transmission mode and perform hypothesis tests for different models of inheritance [64]. As opposed to segregation analysis, CSA can go one step further and work with pedigrees of arbitrary structure instead of nuclear families and with both quantitative and qualitative traits [54].

We perform a kind of segregation analysis but do not test for a specific genetic model. In accordance with the argument in Houle *et al.* [49], we employ a phenotype-based approach to study the inheritance mechanisms, because the details of genetic causation of CRC are

still unknown and complex, and the assumptions of a genotype-based approach may not hold true.

### 3.2.4 Marginalization is a problem

This problem has been approached in previous work done by our group [90]. Because the latent variables are unknown, a straightforward estimation procedure has to marginalize the likelihood respective to each one. The inheritability of this latent variable means that observations within a family are dependent, and the marginalization cannot happen on the level of a single person but over a whole family. Since each latent variable can assume one of two values, the complexity of computing this sum is $\mathcal{O}(2^D)$, where $D$ is the number of family members.

The runtime of this straightforward optimization over the marginalized likelihood is still reasonable when no family has an excessive number of members. However, the number of possible risk constellations within a family grows two-fold with each new family member. As soon as even one family is sufficiently large, the marginalization quickly becomes unfeasible.

In these situations, an alternative approach is needed.

### 3.2.5 Aim of this chapter

We will approach the problem described above in two ways:

First, we describe and implement heuristics to greatly reduce the computation time for the straightforward optimization of the marginalized likelihood. To that end, we reduce the number of iterations in the marginalizing sum as much as possible a priori, and then use the optimization technique of *memoization* [79] within the iterations to evaluate costly functions as few times as possible.

Second, we implement an EM algorithm for situations in which families are too large for the marginalization procedure. The E-step is nontrivial because the latent variables within a pedigree are dependent, and a straightforward calculation of the marginal posteriors would again be of exponential runtime. For a linear dependency structure (e.g., in a Hidden Markov Model), the Baum-Welch algorithm [12] is an efficient method for solving the E-step. In our problem, the data instead shows dependency in a tree structure. This dependency structure necessitates using the sum-product algorithm [62] to obtain the marginalized posterior probabilities in the E-step in linear time. A similar approach for analyzing the marginalization over hidden variables has been proposed and implemented in [34] in the completely different context of single-cell time lapse image analysis.

We show that the runtime of our EM algorithm is linear instead of exponential in terms of the pedigree size. We also execute a simulation study to show that our algorithm correctly recovers the specified parameters.

Finally, we demonstrate the runtime improvement of our algorithm on a real data set: a family study of CRC cases in Upper Bavaria.

Details of the biological relevance of this analysis are discussed elsewhere [90].

## 3.3 Methods

### 3.3.1 Nomenclature

The data set is composed of families that are represented as pedigrees (Figure 3.1). We call individuals at the top of the pedigree (i.e., those with unspecified parents) *founder nodes*, and all other persons are *nonfounders*. Individuals without any offspring (i.e., at the bottom of the pedigree) are called *final* individuals.

We denote by $t_i$ the chronological age in years of onset of CRC for each person $i = 1, \ldots, n$, if the corresponding censoring indicator $c_i$ equals 1, and the age at censoring if $c_i = 0$. The gender of an observation is denoted by $m_i$, which is 1 for males and 0 for females. The observed data for one person is thus $x_i = (t_i, c_i, m_i)$.

Each person also has a latent variable $z_i$, which equals 1 if this person is a risk carrier, and 0 if not. We use $\male_i$ and $\female_i$ to denote the position (i.e., the value of $i$) of the father and mother of person $i$. For example, if we have a risk status $z_i$ for a nonfounder $i$, this individual's father's risk status is $z_{\male_i}$.

We denote the set of all $i$ that are founder nodes by $F$.

The complete data vectors for all patients are called $x$ and $z$, respectively.

### 3.3.2 Penetrance model

For persons where $z_i = 1$, we assume an elevated relative risk of developing CRC, which manifests itself through a hazard rate increased by a multiplicative factor $\alpha$, the *penetrance* [18]. This parameter is unknown and will be estimated.

We assume a Weibull distribution for $t_i$. The Weibull hazard rate is given by $h(t) = k\lambda^k t^{k-1}$, with the parameters $k > 0$ and $\lambda > 0$. In our relative risk model, we multiply the hazard rate by $\alpha$ if $z_i = 1$ and, additionally, by $\beta$ if $m_i = 1$. These factors model the increased relative risk for risk carriers and males, respectively. Our hazard rate for an event (i.e. diagnosis of CRC) is then

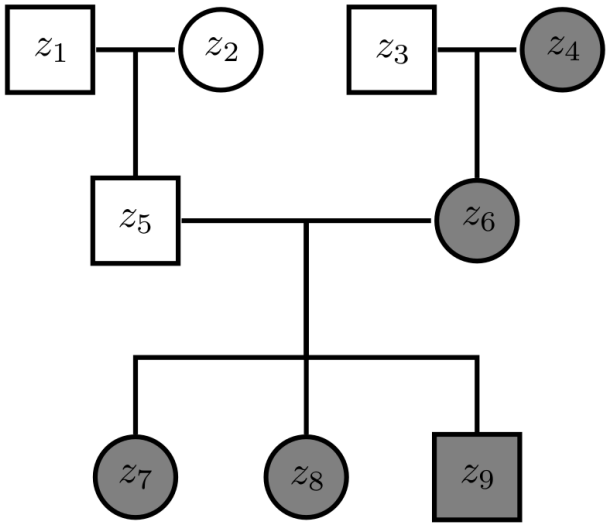



Figure 3.1: A sample pedigree of a family with nine members. Squares denote males, circles females. A couple (a connected circle and square in the same row) gives rise to a set of children (the nodes connected to this couple in the row below). Persons shaded in gray are risk carriers. The four grandparents in the top row are the *founder nodes* in this family, the other five persons are *nonfounders*.

$$h(t_i) = k\lambda^k t_i^{k-1} \alpha^{z_i} \beta^{m_i}$$

The survival function is defined as $S(t) = \exp(-\int_0^t h(u)\mathrm{d}u)$. With the additional relative risk factors, this becomes

$$S(t_i) = \exp(-(t_i\lambda)^k \alpha^{z_i} \beta^{m_i})$$

The density for one observation $i$ is composed of the product of the survival function and (for uncensored observations) the hazard rate:

$$f(t_i|z_i) = h(t_i)^{c_i} \cdot S(t_i)$$

The observations $x_i$ are conditionally independent given $z_i$, and the density of the whole data $f(x|z,\theta)$ can be split up into a product of individual densities: $f(x|z,\theta) = \prod_i f(x_i|z_i,\theta)$.

### 3.3.3 Heritage model

The *founder prevalence* (i.e., the a-priori probability $\mathbb{P}(Z_i = 1)$ for a founder node to carry the risk factor) is called $p_1$. This is the second parameter we will estimate.

The probability for a nonfounder to be a risk carrier is dependent on its parents' risk statuses and the *inheritance probability* $p_H$. Our model does not allow for spontaneous mutations to risk carrier.

If either parent passes down a risk factor $z_{♂_i} = 1$ or $z_{♀_i} = 1$ with the probability $p_H$, then the probability for the offspring to be a risk carrier is

$$\tilde{p}_i = \mathbb{P}(Z_i = 1 | z_{♂_i}, z_{♀_i}) = p_H z_{♂_i} + p_H z_{♀_i} - p_H^2 z_{♂_i} z_{♀_i}, \tag{3.1}$$

We denote $\mathbb{P}(Z_i = 1 | z_{♂_i}, z_{♀_i})$ for nonfounders by $\tilde{p}_i$ to emphasize the distinction from $p_1$ for founders.

A sensitivity analysis found that varying the value of $p_H$ has a negligible effect on the final parameter estimates [90], and thus we chose $p_H = 0.5$ for all our analyses.

Given a pre-defined inheritance probability $p_H$ and a founder prevalence $p_1$, the probability for a risk vector $Z$ for the entire dataset becomes

$$\begin{aligned}
\mathbb{P}(z) &= \prod_{i \in F} \mathbb{P}(z_i) \cdot \prod_{i \notin F} \mathbb{P}(z_i | z_{♂_i}, z_{♀_i}) \\
&= \prod_{i \in F} p_1^{z_i} (1 - p_1)^{1 - z_i} \cdot \prod_{i \notin F} \tilde{p}_i^{z_i} (1 - \tilde{p}_i)^{1 - z_i}
\end{aligned}$$

### 3.3.4 The complete likelihood

All Weibull parameters $(k, \lambda)$ as well as the inheritance probability $p_H$ and the risk increase for males $(\beta)$ are assumed to be known. We set $k = 4$ and $\lambda = 0.0058$ according to previous analyses done by our group [90], $\beta = 2$ according to [59], and $p_H = 0.5$. The complete likelihood where both $x$ and $z$ are observed, is then

$$\begin{aligned}
L(\theta; x, z) = f(x, z) &= \mathbb{P}(z) \cdot f(x | z) \\
&= \prod_{i \in F} \mathbb{P}(z_i) \cdot \prod_{i \notin F} \mathbb{P}(z_i | z_{♂_i}, z_{♀_i}) \cdot \prod_{i=1}^{n} f(x_i | z_i) \\
&= \prod_{i \in F} p_1^{z_i} (1 - p_1)^{1 - z_i} \cdot \prod_{i \notin F} \tilde{p}_i^{z_i} (1 - \tilde{p}_i)^{1 - z_i} \\
&\quad \cdot \prod_{i=1}^{n} [k \lambda^k t_i^{k-1} \alpha^{z_i} \beta^{m_i}]^{c_i} \exp(-(t_i \lambda)^k \alpha^{z_i} \beta^{m_i})
\end{aligned} \tag{3.2}$$

The two factors $f(x|z)$ and $\mathbb{P}(z)$ were defined in the penetrance model and the inheritance model, respectively. The parameter vector in our model is $\theta = (p_1, \alpha)$.

Note that the relevant part for $\alpha$ includes all persons, and the part for $p_1$ only includes the founders. The product over all $i \notin F$ is independent of $\theta$ and thus becomes irrelevant in the estimation procedure.

The complete log-likelihood is then

$$\begin{aligned} l(\theta; x, z) = &(\sum_{i \in F} z_i) \log p_1 + (|F| - \sum_{i \in F} z_i) \log(1 - p_1) + \sum_{i \notin F} [z_i \log \tilde{p}_i + (1 - z_i) \log(1 - \tilde{p}_i)] + \\ &+ \sum_{i=1}^{n} c_i \cdot [\log k + k \log \lambda + (k-1) \log t_i + z_i \log \alpha + m_i \log \beta] - (t_i \lambda)^k \alpha^{z_i} \beta^{m_i} \\ &\text{respective to } \alpha \text{ and } p_1, \text{ this reduces to} \\ &= \text{const} + (\sum_{i \in F} z_i) \log p_1 + (|F| - \sum_{i \in F} z_i) \log(1 - p_1) \\ &\quad + \sum_{i=1}^{n} c_i z_i \log \alpha - (t_i \lambda)^k \alpha^{z_i} \beta^{m_i} \end{aligned} \tag{3.3}$$

If the latent variables $Z$ were actually known, the closed form solutions for $p_1$ and $\alpha$ would be

$$\begin{aligned} \frac{\partial}{\partial p_1} l(\theta; X, Z) &= \frac{\sum_{i \in F} z_i}{p_1} - \frac{|F| - \sum_{i \in F} z_i}{1 - p_1} \\ \hat{p}_1 &= \frac{\sum_{i \in F} z_i}{|F|} \end{aligned} \tag{3.4}$$

and

$$\begin{aligned} \frac{\partial}{\partial \alpha} l(\theta; X, Z) &= \frac{\sum_{i=1}^{n} c_i z_i}{\alpha} - \sum_{i=1}^{n} \mathbb{I}(z_i = 1)(t_i \lambda)^k \beta^{m_i} \\ \hat{\alpha} &= \frac{\sum_{i=1}^{n} c_i z_i}{\sum_{i=1}^{n} \mathbb{I}(z_i = 1)(t_i \lambda)^k \beta^{m_i}} \end{aligned} \tag{3.5}$$

### 3.3.5 The incomplete/marginalized likelihood

We marginalize the non-reduced form of the complete likelihood to obtain the incomplete likelihood $L(\theta; x)$ [97, Eq. 1.5]:

$$L(\theta; x) = \sum_{z} L(\theta; x, z) \tag{3.6}$$

### 3.3.6 Optimizing the marginalized likelihood with Nelder-Mead

To estimate the parameters $p_1$ and $\alpha$, one can use a Nelder-Mead optimization [82] on the marginalized likelihood $L(\theta; x)$.

Unfortunately, for a family of size $D$, the sum over all $z$ has $2^D$ elements. Even when splitting the sum up across all families (see Appendix A.1), the number of summands grows exponentially with increasing family size $D$. Thus, for larger families, the computation of the marginalization within the likelihood evaluation quickly becomes unfeasible.

However, the Nelder-Mead optimization is sufficiently fast, even faster than the EM algorithm, for small families. We therefore implemented two heuristics, grid purging and memoization, to improve the speed of the evaluation of the marginalized likelihood (Equation 3.6) as much as possible.

#### 3.3.6.1 Grid purging

First, we implemented a procedure we call *grid purging*. In it, we adapt methods that have similarly been applied in multipoint linkage analysis [72, 61] that reduce the inheritance space to greatly speed up the computation time.

The procedure described in this section is applied to each family separately and successively. We therefore restrict the data and the respective notation $x$ and $z$ to one family.

For a family of size $D$, we call the $2^D \times D$ matrix of possible $Z$ vectors the *grid* (Table 3.1). During grid purging, we remove all rows of $Z$ where $\mathbb{P}(Z) = 0$, because these constellations result in a summand of 0 in the marginalizing sum of the likelihood. These are invalid risk vectors, where, for example, both parents have a risk status of $Z_i = 0$, but one of their children has $Z_i = 1$. See Figure 3.2 and Table 3.1 for an example of a family with three persons: a father, a mother, and one child.

Thus, when $\mathbb{P}(Z) = 0$, we save time by not computing the density $f(x|z)$ there. By using this procedure, we reduce the number of iterations in the marginalizing sum to a fraction of around 1/7 for large families. In the simpler case of $p_H = 1$ (i.e., a full inheritance

model) the risk status $Z_i$ for all nonfounders is deterministic given the founders, and the sum reduces to $2^F$ iterations.

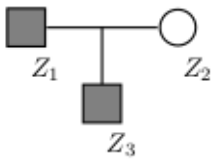



Figure 3.2: A pedigree of a family of three. The gray shading represents a possible risk vector, in this case, $Z_1 = 1$, $Z_2 = 0$, and $Z_3 = 1$.

| $Z_1$ | $Z_2$ | $Z_3$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |
| 1 | 1 | 1 |

Table 3.1: Possible risk vectors for a family of 3 (Figure 3.2). Each row represents one possible constellation of the vector $Z$. The second, gray row is an impossible constellation (our model does not allow for spontaneous mutations) and will be removed in the grid purging procedure.

To create the grid of possible $Z$ vectors, building a $2^D \times D$ grid and then deleting all rows where $\mathbb{P}(Z) = 0$ is costly memory-wise. For larger families, we instead started with an empty "candidates" grid of possible $Z$ vectors. We then expanded one $2^{|F|} \times |F|$ grid over the founders, and one $2^{D-|F|} \times (D - |F|)$ grid over the non-founders (Table 3.2). Subsequently, we looped row-wise over the founders grid. For each row, we then determined the possible vectors from the non-founder grid. These vectors were then added to the candidates grid.

Note that this procedure should be done only once, at the beginning of the data processing. The grids of possible $Z$ should then be stored as an object.

#### 3.3.6.2 Memoization

In a second step, we accelerated the computations within the marginalizing sum by implementing a memoization procedure [79]. Memoization is the process of storing input parameters and return values of a function. If the function is then called a second time

| $Z_1$ | $Z_2$ |
|---|---|
| 0 | 0 |
| 0 | 1 |
| 1 | 0 |
| 1 | 1 |

| $Z_3$ |
|---|
| 0 |
| 1 |

Table 3.2: The two-step procedure used to create the grid of possible $Z$ vectors. We created a nested loop that iterates over the rows of the left table, and then extracts all viable rows from the right table.

with the same arguments, it returns the stored value rather than compute it again. This procedure works for all functions that do not contain stochastic elements (e.g., random number generations).

Because the density and probability calculations within the sum are repeated many times for the same arguments, we pre-compute and store as many values as possible. Specifically, we precompute the grid of the probabilities $\mathbb{P}(Z)$ (see Table 3.1) column-wise. Analogously, we precompute the $2D$ values of the densities $f(x_i|Z_i = 0)$ and $f(x_i|Z_i = 1)$, respectively. In the loop that implements the marginalizing sum, we simply extract and multiply these values.

With these modifications to the evaluation of the marginalized likelihood, we can then finally estimate $p_1$ and $\alpha$ with a Nelder-Mead optimization.

One evaluation of the marginalized likelihood with grid purging and memoization took about 9.2% as long as the evaluation by a straightforward implementation of Equation 3.6. Nevertheless, for large families with more than 20 members, the computation of both the grid purging at the beginning and the marginalization within the likelihood evaluation quickly becomes unfeasible.

### 3.3.7 The EM algorithm

A common approach for finding MLEs in the presence of latent variables is to use the EM algorithm. Resources on the EM algorithm are plentiful, including a short tutorial [27], the seminal paper by Dempster, Laird and Rubin [28], and an entire book [75] devoted to the subject.

In short, the EM algorithm proceeds in a loop over two steps. In the *E-step*, one calculates the expected log-likelihood over the latent variables $Z$, given the observed data and the current parameter estimates. This problem reduces to computing complete-data sufficient statistics [28]. In the subsequent *M-step*, one then updates the estimates of the parameters, given the new expected sufficient statistics from the E-step.

As a convergence criterion, frequent choices include the size of the relative change of either the log-likelihood or the parameter estimates [1]. If the emphasis lies on compliance with a marginalized optimization, implementing and using the function $B(\theta; \theta^{(t)})$ from Dellaert [27] is the better choice, since this function converges to the true likelihood as one approaches the MLE estimates and thus allows a direct comparison between the two methods. However, evaluating $B$ in each iteration is a costly step. Instead, we use the size of the relative change of the parameter estimates for $\alpha$ and $p_1$ as a stopping criterion. This is more conservative than using the log-likelihood [1], and we are on the safe side by letting the algorithm run a bit longer than it would have to.

In the remainder of this chapter, we set the stopping criterion to a relative tolerance of 0.0005, unless stated otherwise.

#### 3.3.7.1 The expected log-likelihood $Q(\theta; \theta^{(t)})$

To compute the expected log-likelihood $Q(\theta; \theta^{(t)})$, we introduce the *membership probabilities* [75, p. 43] $T_i^{(t)}$. This is the probability for *one* person's risk status $Z_i$, given the *whole* observed data $x$:

$$\begin{aligned} T_i^{(t)} &= \mathbb{E}_{Z|x,\theta^{(t)}}(Z_i) \\ &= \mathbb{E}_{Z_i|x,\theta^{(t)}}(Z_i) \\ &= \mathbb{P}(Z_i = 1|x, \theta^{(t)}) \\ &= \sum_z z_i \mathbb{P}(z|x, \theta^{(t)}) \end{aligned} \tag{3.7}$$

The expected value $\mathbb{E}_{Z|x,\theta^{(t)}}(Z_i)$ is equal to the marginalized expected value $\mathbb{E}_{Z_i|x,\theta^{(t)}}(Z_i)$ because of the following marginalization steps from $Z$ to $Z_i$:

$$\begin{aligned} T_i^{(t)} = \mathbb{E}_{Z|x,\theta^{(t)}}(Z_i) &= \sum_z z_i \mathbb{P}(z|x, \theta^{(t)}) \\ &= \sum_{z_1} \cdots \sum_{z_n} z_i \mathbb{P}(z_1, z_2, \ldots, z_n|x, \theta^{(t)}) \\ &= \sum_{z_i} z_i \underbrace{\sum_{z_1} \cdots \sum_{z_{i-1}} \sum_{z_{i+1}} \cdots \sum_{z_n} \mathbb{P}(z_1, z_2, \ldots, z_n|x, \theta^{(t)})}_{=\mathbb{P}(z_i|x,\theta^{(t)})} \\ &= \mathbb{E}_{Z_i|x,\theta^{(t)}}(z_i) \\ &= \mathbb{P}(Z_i = 1|x, \theta^{(t)}) \end{aligned} \tag{3.8}$$

Here, the summation is over all admissible combinations of $z_i$; that is, $\mathbb{P}(z) > 0$ and $z_i = 1$, because all other summands become zero. The condition on the entire observed data $x$ and the summation over all $z$ will conveniently reduce to a condition on and summation of only the respective family's data $x$ and $z$ (Appendix A.1). The target function $Q$ becomes (cf. Equation 3.3)

$$\begin{aligned} Q(\theta;\theta^{(t)}) &= \mathbb{E}_{Z|x,\theta^{(t)}}[l(\theta;x,Z)] \\ &= \text{const} + \log(p_1)(\sum_{i\in F} T_i^{(t)}) + \log(1-p_1)(\sum_{i\in F}(1-T_i^{(t)}))+ \\ &\quad + \sum_{i=1}^{n} T_i^{(t)} c_i \log\alpha - (t_i\lambda)^k \beta^{m_i}(T_i^{(t)}\alpha + (1-T_i^{(t)})) \end{aligned} \tag{3.9}$$

#### 3.3.7.2 The M-step

For the M-step, we maximize $Q(\theta;\theta^{(t)})$ respective to $\alpha$ and $p_1$ to obtain the new parameter estimates for iteration $t+1$. Once the values of all $T_i^{(t)}$ are known, the maximization of $Q$ with respect to $p_1$ and $\alpha$ is straightforward and has a closed form solution:

$$p_1^{(t+1)} = \frac{\sum_{i\in F} T_i^{(t)}}{|F|} \tag{3.10}$$

$$\alpha^{(t+1)} = \frac{\sum_{i=1}^{n} c_i T_i^{(t)}}{\sum_{i=1}^{n} T_i^{(t)}(t_i\lambda)^k \beta^{m_i}} \tag{3.11}$$

Note their similarity to Equations 3.4 and 3.5.

#### 3.3.7.3 The E-step

It follows from Equations 3.9 and 3.10 that, as in the "standard" examples of the EM algorithm, the E-step conveniently reduces to computing the complete-data sufficient statistics $T_i^{(t)}$.

The reason for this simplification is that the log-likelihood is linear in the latent data $Z$. Computing $Q(\theta;\theta^{(t)})$ thus simplifies to replacing each occurring $Z_i$ with its conditional expectation $T_i^{(t)}$. The M-step then uses these "imputed" values of the latent data $Z$ for the updated parameter estimates.

#### 3.3.7.4 Marginalization of the joint density

In our setting, the difficulty in computing $T_i^{(t)}$ is that the probability for the risk status of one family member $Z_i$ is conditioned on the observed data $x$ of the *entire family* (Appendix A.1). To compute these values, we would have to marginalize over all risk vectors $z$ where $z_i = 1$, i.e. $T_i^{(t)} = \sum_r \mathbb{P}(Z = r|x, \theta^{(t)})$, where $r$ is a valid risk vector (i.e. with $\mathbb{P}(Z = r) > 0$ and with $z_i = 1$). We would end up with the same exponential runtime as in a Nelder-Mead optimization.

Alternatively, a pedigree can be represented as a Bayesian network [85, 4], also known as a causal probabilistic network (CPN), which in turn can be converted into a factor graph [62]. This representation is advantageous because it allows the efficient computation of marginals via the sum-product algorithm.

The sum-product algorithm [62], also known as the belief propagation algorithm, computes marginalizations of the form of $T_i^{(t)}$ in linear runtime [78, p. 290]. It does this by representing a complex "global" function $g(z)$ – here, $f(x, z|\theta^{(t)})$ – as a factor graph; that is, a product of multiple "local" functions, $\prod_j \phi_j$, each depending on only a subset of the arguments in $g(z)$.

The sum-product algorithm then exploits this structure to efficiently compute marginalizations of $g(z)$. Here, we marginalize the joint density to obtain $f(z_i, x|\theta^{(t)})$. By dividing this joint density through $f(x) = f(Z_i = 1, x|\theta^{(t)}) + f(Z_i = 0, x|\theta^{(t)})$, we ultimately obtain $T_i^{(t)} = \mathbb{P}(Z_i = 1|x, \theta^{(t)})$, which was our actual goal.

### 3.3.8 Factor graphs

Factor graphs were first introduced by Kschischang [62] to represent factorizations of multivariate functions. To avoid confusion between factors and density functions $f$, we deviate from the notation of Kschischang and denote factor nodes with $\phi$ instead of $f$.

The factor graph in Figure 3.3 encodes the joint density $g(z) = f(z, x)$ of the family from Figure 3.1 as the product of 7 factors $\phi_j$:

$$\begin{aligned} g(z) = {} & \phi_1(z_1) \cdot \phi_2(z_2) \cdot \phi_3(z_3) \cdot \phi_4(z_4) \\ & \cdot \phi_5(z_1, z_2, z_5) \cdot \phi_6(z_3, z_4, z_6) \cdot \phi_{789}(z_5, z_6, z_7, z_8, z_9) \end{aligned} \tag{3.12}$$

As an example, the factors $\phi_j$ from Figure 3.3 describe the following functions:

$$\phi_1(z_1) = f(x_1|z_1)\ \mathbb{P}(z_1)$$

$$\phi_2(z_2) = f(x_2|z_2)\ \mathbb{P}(z_2)$$

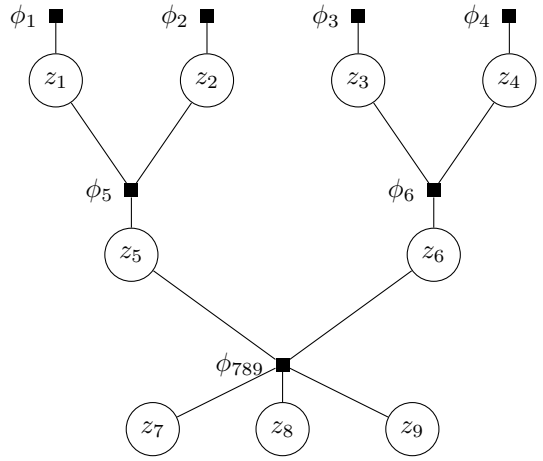



Figure 3.3: A factor graph visualizing the factorization of $g(z) = f(x, z)$ (Equation 3.2) for the family from Figure 3.1. Circles represent variable nodes, and filled squares represent factor nodes, i.e. local functions. The edges show which variables are arguments to which factors. For example, the factor $\phi_6$ has three arguments: $\phi_6(z_3, z_4, z_6)$

$$\phi_3(z_3) = f(x_3|z_3)\ \mathbb{P}(z_3)$$

$$\phi_4(z_4) = f(x_4|z_4)\ \mathbb{P}(z_4)$$

$$\phi_5(z_1, z_2, z_5) = f(x_5|z_5)\ \mathbb{P}(z_5|z_1, z_2)$$

$$\phi_6(z_3, z_4, z_6) = f(x_6|z_6)\ \mathbb{P}(z_6|z_3, z_4)$$

$$\phi_{789}(z_5, z_6, z_7, z_8, z_9) = f(x_7|z_7)\ \mathbb{P}(z_7|z_5, z_6)\cdot f(x_8|z_8)\ \mathbb{P}(z_8|z_5, z_6)\cdot f(x_9|z_9)\ \mathbb{P}(z_9|z_5, z_6).$$

The factor $\phi_{789}$ cannot be split up into three factors because the graph edges would then form a *cycle*, which is not allowed, or it would necessitate a costly *loopy belief propagation* procedure [62, 52]. Instead, we implement a *clustering* procedure [62] and group the respective densities into one factor per set of parents. As we will see later, marginalizing this expression still runs in linear time.

In general, the factors are defined as

$$\phi_J(z_J, z_{\male_J}, z_{\female_J}) = \prod_{j \in J} f(x_j|z_j)\mathbb{P}(z_j|z_{\male_j}, z_{\female_j}),$$

where $J$ is the set of all children with the same parents, which are denoted by $z_{\male_J}$ and $z_{\female_J}$. If $\phi_J$ is a factor for a founder node, then $z_{\male_J}$ and $z_{\female_J}$ are defined as an empty set and the respective probability $\mathbb{P}(z_j)$ is unconditioned.

### 3.3.9 The sum-product algorithm

Having set up a factor graph for each family, we can then apply the sum-product algorithm to compute marginalizations of $f(z, x)$ at each variable node $z_i$, i.e. $f(z_i, x)$.

In our setting, we restrict ourselves to family *trees*, that is, we do not allow for consanguineous marriages, which would again lead to cycles in the corresponding factor graph.

#### 3.3.9.1 Description of the sum-product algorithm

Let $\mu_{z\to\phi}(z)$ denote the message sent from a variable node $z$ to a factor node $\phi$, and let $\mu_{\phi\to z}(z)$ denote the message sent from a factor node $\phi$ to a variable node $z$. Furthermore, let $n(v)$ denote the set of neighboring nodes of a (factor or variable) node $v$.

We then define the messages from a variable node to a factor node, and from a factor node to a variable node, as follows [62]:

$$\mu_{z\to\phi}(z) = \prod_{h\in n(z)\setminus\{\phi\}} \mu_{h\to z}(z)$$

$$\mu_{\phi\to z}(z) = \sum_{\sim\{z\}} \left( \phi(Z_\phi) \prod_{y\in n(\phi)\setminus\{z\}} \mu_{y\to\phi}(y) \right)$$

where $Z_\phi$ is the set of arguments of the factor $\phi$. If $h \in n(z) \setminus \{\phi\} = \{\emptyset\}$, for example, at a variable node of a final individual (the grandchildren $z_7$, $z_8$, and $z_9$ in Figure 3.3), the product is defined as 1. The expression $\sum_{\sim\{z\}}$ is adapted from Kschischang et al. [62] and denotes the *not-sum*, that is, the sum over all variables except $z$.

Finally, the marginalization, or *termination step*, computes the value of $g_i(z_i) = \sum_{\sim\{z_i\}} g(z)$ as the product of all incoming messages on a variable node $z_i$. The marginalized $g_i(z_i)$ are equal to $f(z_i, x|\theta^{(t)})$, that is, they are proportional to the desired outputs $T_i^{(t)}$ from the E-step in the EM algorithm.

#### 3.3.9.2 Some example messages and marginalizations of the sum-product algorithm

We illustrate the sum-product algorithm by calculating two example messages and one example marginalization from the pedigree in Figure 3.3.

First, the message $\mu_{z_6 \to \phi_{789}}(z_6)$ from the variable node $z_6$ to the factor node $\phi_{789}$ equals

$$\mu_{z_6 \to \phi_{789}}(z_6) = \mu_{\phi_6 \to z_6}(z_6)$$

Second, the message $\mu_{\phi_5 \to z_2}(z_2)$ from the factor node $\phi_5$ to the variable node $z_2$ equals

$$\mu_{\phi_5 \to z_2}(z_2) = \sum_{z_1} \sum_{z_5} (\phi_5(z_1, z_2, z_5) \cdot \mu_{z_1 \to \phi_5}(z_1) \mu_{z_5 \to \phi_5}(z_5))$$

Last, we compute the example marginalization at the variable node $z_5$ as

$$g_5(z_5) = f(z_5, x|\theta^{(t)}) = \mu_{\phi_5 \to z_5}(z_5) \cdot \mu_{\phi_{789} \to z_5}(z_5)$$

Then,

$$\begin{aligned} T_5^{(t)} &= \mathbb{P}(Z_5 = 1|x, \theta^{(t)}) \\ &= \frac{f(z_5 = 1, x|\theta^{(t)})}{f(z_5 = 0, x|\theta^{(t)}) + f(z_5 = 1, x|\theta^{(t)})} \\ &= \frac{\mu_{\phi_5 \to z_5}(1) \cdot \mu_{\phi_{789} \to z_5}(1)}{\mu_{\phi_5 \to z_5}(0) \cdot \mu_{\phi_{789} \to z_5}(0) + \mu_{\phi_5 \to z_5}(1) \cdot \mu_{\phi_{789} \to z_5}(1)} \end{aligned}$$

This shows that the desired values $T_i^{(t)}$ from the E-step are immediately obtained as soon as all possible messages are computed.

Appendix A.2 and Figure A.1 show the detailed derivation of all remaining messages and illustrate the stepwise procedure to obtain all messages.

### 3.3.10 Convergence of the EM algorithm

A common drawback among many estimation algorithms is the danger of converging into a local maximum. This also applies to the EM algorithm. However, we will show that the marginal likelihood function (Equation 3.6) is concave and thus the EM algorithm will always converge to the global maximum of the marginal likelihood, irrespective of the initial parameter choice.

A twice differentiable function is concave if its Hessian matrix is negative semidefinite. We first show that the Hessian of the complete log-likelihood function (Equation 3.3) is

negative semidefinite. Since the variables $\alpha$ and $p_1$ are separated, $\frac{\partial^2}{\partial\alpha\partial p_1} l(\theta; x, z) = 0$, and consequently the Hessian is a diagonal matrix. It suffices to show that both its diagonal entries are zero or negative. We obtain

$$\frac{\partial^2}{\partial p_1^2} l(\theta; x, z) = -\frac{\sum_{i\in F} z_i}{p_1^2} - \frac{|F| - \sum_{i\in F} z_i}{(1-p_1^2)} \tag{3.13}$$

$$\frac{\partial^2}{\partial \alpha^2} l(\theta; x, z) = -\sum_{i=1}^{n} \frac{c_i z_i}{\alpha^2} \tag{3.14}$$

Note that in Equation 3.13, $\sum z_i \leq |F|$, hence both terms on the right hand side of this equation are negative or zero.

This proves that the complete log-likelihood is concave. Since the exponential is strictly monotonically increasing and $l(\theta; x, z)$ is strictly concave, it follows that $L(\theta; x, z) = \exp(l(\theta; x, z))$ has only one unique local and hence global maximum [20]. Finally, the marginal log-likelihood, as the sum $\sum_z l(\theta; x, z)$ of concave functions, is also concave.

This means that simpler but faster optimization algorithms such as Nelder-Mead can be used for our model, and there is no need for more complex algorithms such as stochastic or constrained optimizers (e.g., L-BFGS-B [23] or simulated annealing [14]).

### 3.3.11 Application: Estimating the probability of being a risk family

Applying the sum-product algorithm directly implies a straightforward method to compute the probability of being a risk family.

After we have estimated $p_1$ and $\alpha$, we can estimate the probability that a family carries the risk factor for a new pedigree (or a pedigree from the original study, i.e. the training data). We define a *risk family* as a family in which at least one member is carrying the risk factor; that is, $z_i = 1$ for at least one $i$. This is exactly one minus the probability that *no* family member carries the risk factor. If we restrict the data $x$ and $Z$ to the family in question, and define the event $R$ as "The family is a risk family", we can then compute

$$\mathbb{P}(R) = 1 - \mathbb{P}(Z = 0|x, \hat{\theta}) \tag{3.15}$$

$$= 1 - \prod_{i\in F} \mathbb{P}(Z_i = 0|x, \hat{\theta}) \tag{3.16}$$

$$= 1 - \prod_{i\in F} (1 - T_i^{(t)}) \tag{3.17}$$

The step from Equation 3.15 to Equation 3.16 is possible because the probability that *no family member* carries the risk factor equals the probability that *no founder* carries the risk factor, because the former is true if and only if the latter is true. Then, we can split up the joint probability that no founder carries the risk factor into the individual probabilities $\mathbb{P}(Z_i = 0|x, \hat{\theta})$. This step is possible because we consider only founders, and their risk probabilities are independent of any other $z_i$.

Since $\mathbb{P}(Z_i = 0|x, \hat{\theta})$ equals $1 - T_i^{(t)}$ by definition (Equation 3.7), we can simply run the E-step of the EM algorithm once on the new family to obtain these values. We then multiply over only those $T_i^{(t)}$ where $i \in F$ and obtain $\mathbb{P}(R)$, an estimator for the familial CRC risk.

Therefore, the sum-product algorithm also conveniently simplifies the subsequent step after the parameter estimation, that is, the estimation of being a risk family. Without it, the conditioning on the whole family's observed data $x$ would again necessitate a marginalization of exponential complexity.

### 3.3.12 A general sum-product algorithm for pedigrees in R

We implemented a sum-product algorithm for computing the marginals of an arbitrary pedigree in R [88] and made it available on GitHub. The code creates one factor graph per family, and therein one factor node per founder, which contains $\phi_i(z_i) = f(x_i|z_i) \cdot \mathbb{P}(z_i)$. Furthermore, we create one factor per set of parents, which contains the product of the densities of all children (but not the parents): $\phi_j(Z_j) = \prod_{i \in K_j} f(x_i|z_i) \cdot \mathbb{P}(z_i|z_{♂_i}, z_{♀_i})$, where $Z_j$ represents all variables within the factor (parents and children), and $K_j$ is the set of children variables connected to $\phi_j$.

Messages from a "large" factor containing parents and many children will be summed over all neighboring variable nodes except the destination variable node. By iteratively exploiting the distributive law, this sum can be efficiently broken down from exponential to linear runtime. We illustrate the procedure based on an example:

Consider from Figure 3.3 the message

$$\mu_{\phi_{789} \to z_8}(z_8) = \sum_{z_5=0}^{1} \sum_{z_6=0}^{1} \sum_{z_7=0}^{1} \sum_{z_9=0}^{1} \phi_{789}(z_5, z_6, z_7, z_8, z_9) \cdot \mu_{z_5 \to \phi_{789}}(z_5) \mu_{z_6 \to \phi_{789}}(z_6) \mu_{z_7 \to \phi_{789}}(z_7) \mu_{z_9 \to \phi_{789}}(z_9)$$

Because we can decompose $\phi_{789}(z_5, z_6, z_7, z_8, z_9)$ into the product $f(x_7|z_7)\mathbb{P}(z_7|z_5, z_6) \cdot f(x_8|z_8)\mathbb{P}(z_8|z_5, z_6) \cdot f(x_9|z_9)\mathbb{P}(z_9|z_5, z_6)$, the quadruple sum can be split up into

$$\mu_{\phi_{789}\to z_8}(z_8) = \sum_{z_5=0}^{1}\sum_{z_6=0}^{1}\Bigg\{\mu_{z_5\to\phi_{789}}(z_5)\cdot\mu_{z_6\to\phi_{789}}(z_6)\cdot f(x_8|z_8)\cdot\mathbb{P}(z_8|z_5,z_6) \cdot\left[\sum_{z_7=0}^{1}\mu_{z_7\to\phi_{789}}(z_7)f(x_7|z_7)\mathbb{P}(z_7|z_5,z_6)\right]\left[\sum_{z_9=0}^{1}\mu_{z_9\to\phi_{789}}(z_9)f(x_9|z_9)\mathbb{P}(z_9|z_5,z_6)\right]\Bigg\}$$

This representation of the marginalizing sum can now be evaluated in linear time respective to the number of children.

## 3.4 Results

### 3.4.1 Simulation study

We performed an *in silico* experiment by simulating data sets with a given $p_H$, $p_1$ and $\alpha$ and with a varying number of families ($N$) and pedigree size ($D$). The risk status $z_i$ for each founder was randomly sampled with the probability $\mathbb{P}(z_i = 1) = p_1$, the statuses for all nonfounders were sampled according to Equation 3.1.

The age of onset of CRC was then simulated according to a Weibull distribution with the best fitting parameters according to [90], $\lambda = 0.0058$ and $k = 4$, and a risk increase for males of $\beta = 2$:

$$f(t_i|z_i) = h(t_i)\cdot S(t_i) = [k\lambda^k t_i^{k-1}\alpha^{z_i}\beta^{m_i}]\cdot\exp(-(t_i\lambda)^k\alpha^{z_i}\beta^{m_i})$$

We then simulated a censoring age $u_i$ from the following Gaussian distribution:

$$u_i \sim \mathcal{N}(125, 100)$$

The rather optimistic mean censoring age of 125 years was chosen to keep the ratio of censored subjects below 66%, since a higher censoring rate would just require a larger simulated data set to reach the same stability.

Each subject's censoring indicator $c_i$ was then set to 1 if $t_i < u_i$ and 0 otherwise. A value of 0 therefore indicates a censored observation. If a subject is censored, $t_i$ was replaced by $u_i$, the age at censoring.

#### 3.4.1.1 Runtime improvement

We simulated data sets with different pedigree sizes to investigate the threshold pedigree size from which the EM algorithm is faster than a Nelder-Mead optimization. The pedigrees used were the following:

- $D = 15$: Four generations (8 founders) with one final individual
- $D = 17$: The same pedigree as for $D = 15$, with one additional parent pair for the first founder
- $D = 19$: One more parent pair in the same generation as for $D = 17$
- $D = 21$: One more parent pair in the same generation as for $D = 19$
- $D = 23$: One more parent pair in the same generation as for $D = 21$
- $D = 25$: One more parent pair in the same generation as for $D = 23$
- $D = 31$: Five generations with one final individual

Figure 3.4 and Table 3.3 show the runtime of the Nelder-Mead optimization versus the EM algorithm for different data sizes and pedigree sizes. The results suggest that using the EM algorithm is advantageous as soon as *some* families in the data set are large (i.e., more than around 20 members). For $D = 31$, we extrapolated the runtime for the Nelder-Mead optimization to about 50 days, according to a log-linear regression of runtime against family size. This shows the dramatic improvement of using the EM algorithm as family sizes grow larger. A more advanced EM algorithm can even split the data into small and large pedigrees, and in the E-step use the sum-product algorithm for the larger families, and a "brute force" marginalization for smaller families. We implemented this possibility in our script. In the simulation studies, however, we did not use it to maintain comparability.

| N | 15 | 17 | 19 | 21 | 23 | 25 |
|---|---|---|---|---|---|---|
| 50 | 0.10 | 0.18 | 0.51 | 1.62 | 6.54 | 25.99 |
| 100 | 0.09 | 0.16 | 0.47 | 1.55 | 5.44 | 24.84 |
| 150 | 0.12 | 0.13 | 0.38 | 1.23 | 6.13 | 22.86 |
| 200 | 0.10 | 0.14 | 0.46 | 1.79 | 6.96 | 38.11 |
| 250 | 0.14 | 0.22 | 0.70 | 4.32 | 12.81 | 39.14 |

Table 3.3: Runtime ratio (Nelder-Mead over EM algorithm) over different family sizes $D$ (columns) and different number of families $N$ (rows). The EM algorithm is faster for pedigrees of size 19 and above, regardless of the number of families in the data set.

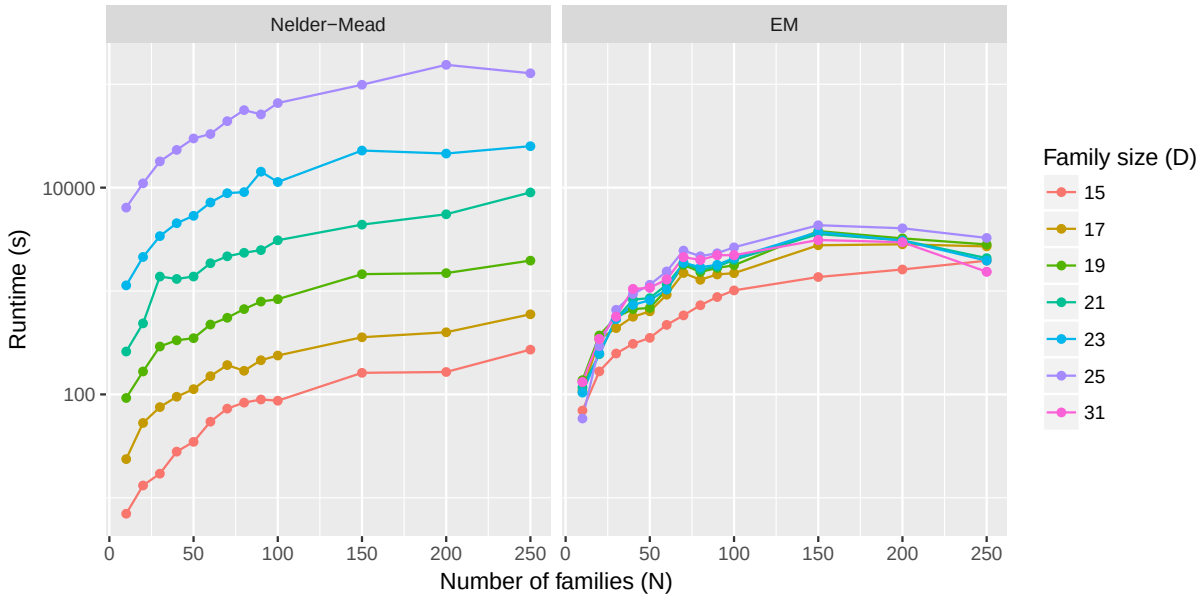



Figure 3.4: Runtime comparison of Nelder-Mead optimization (left) and the EM algorithm (right). Shown is the runtime in seconds on the $y$-axis (log-scale) versus the number of families on the $x$-axis. The effect of an increasing family size $D$ is negligible with the EM algorithm, but exponential with the Nelder-Mead optimization. For the largest families of D=31, only the EM algorithm could be run in a feasible time. The Nelder-Mead optimization would have taken around 50 days.

#### 3.4.1.2 Our algorithm recovers true parameters

We simulated 100 replicated data sets of 500 families of 9 persons as in Figure 3.1. In each replication, we chose $p_1 = 0.2$ and $\alpha = 4$ as the parameters and let the Nelder-Mead optimization and the EM algorithm estimate the parameters to investigate their level of agreement. For this analysis, the stopping criterion (Section 3.3.7) was set to a relative tolerance of 0.00005. Figure 3.5 shows scatterplots and Bland-Altman plots to compare the two methods and finds a strong agreement between them. Table 3.4 shows summary statistics on both methods' parameter estimates in the 100 replications.

| | Minimum | 1st Quartile | Median | Mean | 3rd Quartile | Maximum |
|---|---|---|---|---|---|---|
| $\hat{p_1}$, N-M | 0.1478 | 0.1740 | 0.1913 | 0.1929 | 0.2078 | 0.2864 |
| $\hat{p_1}$, EM | 0.1476 | 0.1739 | 0.1912 | 0.1929 | 0.2078 | 0.2865 |
| $\hat{\alpha}$, N-M | 2.888 | 4.036 | 4.347 | 4.310 | 4.656 | 5.297 |
| $\hat{\alpha}$, EM | 2.890 | 4.037 | 4.345 | 4.310 | 4.660 | 5.287 |

Table 3.4: Five-point summary and mean values for the parameter estimates of the Nelder-Mead optimization (N-M) and the EM algorithm (EM), based on 100 simulated data sets. The simulation parameters were $p_1 = 0.2$ and $\alpha = 4$.

#### 3.4.1.3 The imputation of noninformative parents works

In the real data set, pedigrees were not always recorded in a directly usable manner. For observations with only one available parent, we imputed the missing parent as noninformative ($c_i = 0,\ t_i = 0$ and with the appropriate gender $m_i$). In cases in which a family consisted only of siblings, we imputed both parents as noninformative observations to indicate the relatedness of the siblings.

To assess the feasibility of this procedure, we randomly removed 20% of the family members from the data set after simulation. The simulation and estimation was otherwise performed as in Figure 3.5, just with some family members removed beforehand. The preprocessing then performed an imputation of missing members. The imputation procedure did not affect the results noticeably. Figure 3.6 shows that after our imputing procedure, both algorithms still recover the true parameters.

#### 3.4.1.4 Application: Estimating the probability of being a risk family

We computed the posterior probability of being a CRC risk family for a simulated data set of 1000 pedigrees with 9 persons each, according to Equation 3.17. The resulting ROC curve is shown in Figure 3.7. The AUC of 0.76 shows that risk families can be identified with a satisfyingly good rate.

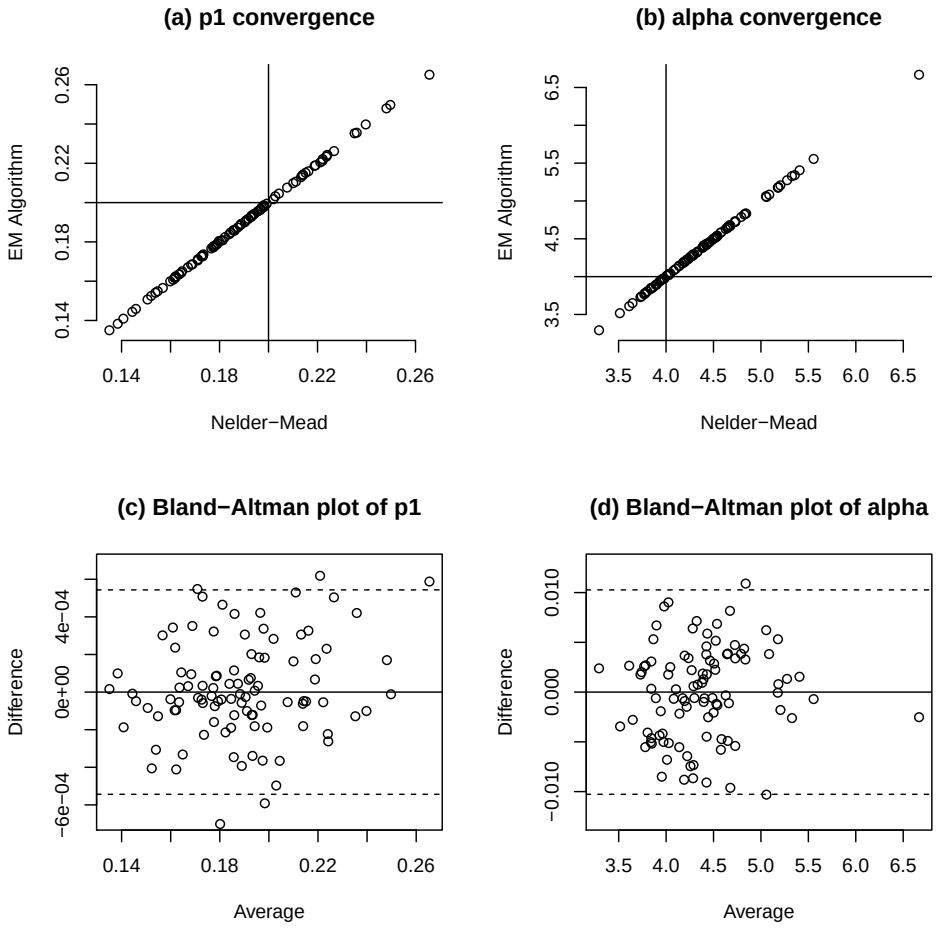



Figure 3.5: Convergence of 100 replications of simulating and estimating data sets. Each replication used random uniform distributed starting values for $\theta = (p_1, \alpha)$. Figures (a) and (b) show the final parameter estimates for the Nelder-Mead optimization ($x$-axis) and the EM algorithm ($y$-axis). Figures (c) and (d) show Bland-Altman plots where the $x$-axis shows the average of the parameter estimates of the two methods, and the $y$-axis shows their difference. Horizontal dashed lines are drawn at $\pm 2$ standard deviations of the difference. We see that both estimation methods agree with each other and converge close to the correct result of $p_1 = 0.2$ and $\alpha = 4$ regardless of starting values.

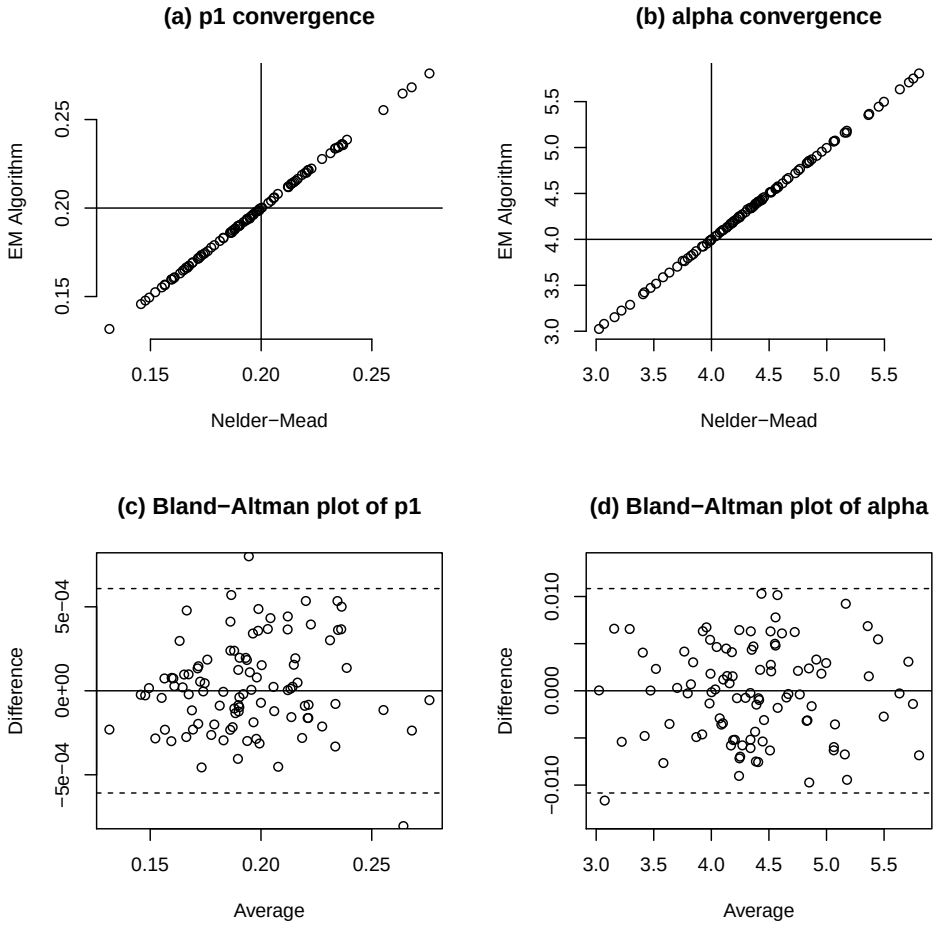



Figure 3.6: Convergence of 100 iterations of imputed data sets. Simulation and estimation was performed as in Figure 3.5, but 20% of the family members were randomly removed beforehand. The preprocessing thus performed an imputation of missing members. The figures show that after our imputing procedure, both algorithms still recover the true parameters.

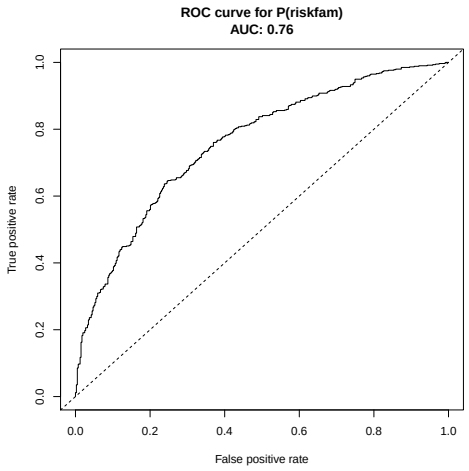



Figure 3.7: An ROC curve for the probability of being a risk family, based on 1000 simulated families with 9 persons (cf. Figure 3.1).

### 3.4.2 Real data

We applied our algorithm to a family study of CRC [71]. In this study, patients diagnosed with CRC in the Munich region were recruited. Subsequently, each of these *index patients* was given a questionnaire with a blank pedigree to fill out data about all known relatives. With this obtained pedigree, the Munich Cancer Registry (MCR) [96] was consulted via an anonymized record-linkage procedure for any CRC diagnoses of the index patient's relatives [81]. The result was a pedigree of family data and CRC diagnoses per index patient, where missing parents were imputed as described in Section 3.4.1.3.

The study was active from September 2012 until June 2014 and resulted in a data set of 611 families, of which 181 were individuals (i.e., a "pedigree" with only one person).

#### 3.4.2.1 Parameter estimates for the real data set

We estimated a prevalence of $p_1 = 0.901$ and a risk factor increase of $\alpha = 5.723$. This rather high a priori probability may stem from a bias in the data set, since the collection procedure preferably selected patients and families that are already exposed to risk. A more detailed discussion on the results is given in [90].

We then performed a 100-fold bootstrapping of families to obtain bootstrap standard errors. The values obtained were $\hat{\sigma}_{p_1} = 0.0002$ and $\hat{\sigma}_\alpha = 0.0229$.

#### 3.4.2.2 Runtime

The data set contained three families with at least 20 members. For the largest family of 23 persons, using the EM algorithm with the sum-product algorithm instead of a Nelder-Mead optimization showed a reduction of the runtime to 24%.

When using the sum-product algorithm for every family, the runtime was 15 times longer than a Nelder-Mead optimization, due to the presence of many families with very few members.

## 3.5 Discussion

Directly translating mathematical formulas into computer code often results in formally correct but slow solutions. In our case, a Nelder-Mead optimization would need multiple evaluations of the marginalized likelihood, which is unfeasible for larger families. An approach based on *peeling* [31, 24, 15], however, could have been used to reduce the time for evaluating the likelihood. The disadvantage of using the Elston-Stewart peeling algorithm is that as soon as the pedigree contains loops, its runtime increases exponentially with the *cutset* (i.e., the number of members that have to be considered jointly) [97]. The EM algorithm coupled with the sum-product algorithm can be extended to pedigrees with loops by applying the "loopy belief propagation" procedure [62]. Furthermore, peeling algorithms need to find an optimal peeling order for each pedigree, a problem that still has no gold standard solution [15]. The sum-product algorithm, on the other hand, directly implies an efficient order of computing the messages, and thus elegantly circumvents this problem. Thompson *et al.* [98] showed that the EM algorithm is a viable alternative to the peeling algorithm in polygenic models. Our approach differs from this in that we skip the detection of responsible genes and instead focus on estimating a family's probability of carrying an (unspecified) CRC risk factor.

The EM algorithm with an approximating Monte Carlo implementation of the E-step has previously been used on pedigrees for segregation analysis [43]. We saw that the EM algorithm in our setting relied on a marginalization over all possible risk vectors for each pedigree. This problem of calculating marginal densities in hierarchical data such as pedigrees has usually been tackled by Markov Chain Monte Carlo (MCMC) simulations and related sampling algorithms [36, 38]. Due to the random sampling, these methods all yield only approximate solutions and may take a long time to reach stable results. Instead, we use the sum-product algorithm [62] within the EM algorithm to solve the necessary marginalization in the E-step in linear instead of exponential time. This provides a fast and exact solution and allows maximum-likelihood estimation with pedigrees of arbitrary size, that furthermore is not dependent on an arbitrarily chosen number of MCMC simulations.

In contrast to complex segregation analysis, our approach is less specific. In particular, we model only an unspecific risk "component", not necessarily a gene or multiple genes, that is passed down to offspring with a certain predefined probability. We chose this phenotype-based approach since the causes for familial occurrence of CRC are currently unknown [49].

Our generic model is therefore not fully in line with Mendelian transmission models. Only under the assumption of an autosomal dominant risk factor that is rare, so that an affected individual can be assumed to have the genotype Aa instead of AA, is a constant inheritance probability of $p_H = 0.5$ justifiable in a Mendelian setting. As an advantage, our approach can model environmental risk factors such as nutrition, lifestyle, or place of residence by choosing $p_H = 1$. If one wants to account for a small probability of changing the place of residence or a probability for children to not adopt the parents' lifestyle, inheritance probabilities of less than 1 can be used as well. However, if a large enough data set were available, it should also be possible to estimate $p_H$ robustly enough.

One limitation of the real data set in this study is the relatively small sample size. With around 600 families, the data set was not large enough to obtain stable estimates. However, because the focus of this study was methodological, the data set could still be used to show the runtime improvement of our algorithm. The moderate gain in speed in the application to the real data set can be explained by two facts: First, the multiplicative time constant for the sum-product algorithm is larger than that for the calculation of the marginal likelihood, since the message passing needs more elementary operations than simple summation over all possible configurations of the family members (yet, as demonstrated in Figure 3.4, this disadvantage is soon compensated by the exponential increase in runtime of the Nelder-Mead method). Second, the families in the real data set are mostly small (median size: 6 members), with the exception of a few larger families (maximum size: 23). Thus, the full power of the EM algorithm will unfold in applications where more larger families are investigated. The size of the families in such studies is likely to grow in the future, with the availability of more longitudinal data. Moreover, note that a single, large family of more than, say, 25 members, will already prohibit the straightforward application of the marginalization approach.

It should also be noted that for small families, the linear runtime of the sum-product algorithm is *slower* than the exponential runtime of the marginalization, due to the overhead in setting up the factor graph (Figure 3.4). In our analysis, the sum-product algorithm had significant benefits only after a family consisted of more than 20 members.

It is possible to estimate further parameters in this model, such as the shape parameter $k$. However, in our case these parameters were external epidemiological information, and thus already available from more robust previous studies. Moreover, if additional parameters are estimated, it should then be reevaluated whether the Hessian matrix is still negative semi definite, i.e. whether the log-likelihood is still concave and unimodal.

Our algorithm can of course also be extended to other models. For example, other pen-

etrance models can be used, such as a "time shift" model, where the hazard rate is not multiplied by a factor $\alpha$, but instead shifted horizontally, by adding a "risk advancement" of a specific number of years [22]. It is also possible to use response distributions other than a Weibull distribution. Furthermore, it is possible to extend our method to model true Mendelian transmission, either by having $Z_i \in \{0, 1, 2\}$ model the number of affected alleles, or by specifying *two* latent variables, $Z_{\male} \in \{0, 1\}$ and $Z_{\female} \in \{0, 1\}$ per individual, that is, one for each allele. One would then need a transmission matrix to specify the probability of each possible outcome for offspring given the statuses of both parents. Ghahramani [39] provides a tutorial on how to extend a Bayesian Network such as a pedigree to deal with multiple latent variables.

When the number of possible genotypes (i.e., the number of possible values for $Z_i$) increases, both the EM algorithm and the Nelder-Mead optimization suffer from exponential runtime increases. This is the case, for example, when one works with multilocus genotypes. The runtime of the EM algorithm is only linear respective to the family size. However, since the genetic mechanism in our case is unknown, a dichotomized latent variable served our purpose well.

Faster algorithms such as the one presented in this chapter also open the door for new analyses that were previously unfeasible. With the sum-product algorithm, we can now conduct large-scale simulation studies for power and sample size determination, and extract further information such as bootstrap confidence intervals from the data.

## 3.6 Conclusion

In this chapter, we developed an efficient algorithm for maximum likelihood estimation where the observations in the data are partially dependent.

The rising size and complexity of modern data sets make it necessary to revisit popular algorithms for data analysis and develop improvements in their efficiency. Here, we considered clinical data in the form of pedigrees, where the presence of latent and inheritable genetic risk factors greatly complicated the analysis procedure.

First, we implemented two heuristics for accelerating a Nelder-Mead optimization of a marginalized likelihood, grid purging and memoization, but the order of the runtime was still exponential. A standard implementation of the EM algorithm results in a runtime that is also still exponential regarding the family sizes, due to the inheritability of the latent variables.

However, by considering the pedigree as a Bayesian network, and then factorizing it with a factor graph and reformulating the E-step by employing a sum-product algorithm, the runtime could be reduced to linear in terms of the family size.

Similar to the peeling algorithm [31], the sum-product algorithm in essence breaks down

complex pedigrees into *nuclear* families, each consisting of father, mother, and all children. The number of children does not cause exponential growth of runtime because the summation is again broken down between each child.

In conclusion, the combination of an EM algorithm with the sum-product algorithm removes the restrictions that exponential runtime imposes on the analysis due to large families, and opens the door for maximum likelihood estimation on large pedigrees.

A useful next step would be to make this risk prediction algorithm available as a web interface so that clinicians can conveniently enter a family's pedigree. It will then aid in assessing the familial CRC risk of individual patients.

# Chapter 4

# Genome-wide Generalized Additive Models

## 4.1 Chapter summary

Chromatin immunoprecipitation followed by deep sequencing (ChIP-Seq) is a widely used method to study protein-DNA interactions. Despite being a relatively new technique, there is currently a multitude of software available, but most of these packages are dedicated to specific applications and are based on mutually incompatible statistical assumptions. To analyze ChIP-Seq data, practitioners are currently required to use a combination of several of these dedicated tools for tasks such as calling protein occupancy peaks or testing for differential occupancies. We therefore aimed to develop a general framework for ChIP-Seq analysis that can be applied to the entire analysis workflow, from signal extraction to the analysis of factorial designs.

Here, we present Genome-wide Generalized Additive Models (GenoGAM), which brings the well-established and flexible generalized additive models framework to genomic applications using a data parallelism strategy. We model ChIP-Seq read count frequencies as products of smooth functions along chromosomes. Smoothing parameters are estimated from the data eliminating the ad hoc binning and windowing needed by current approaches. We derived a peak caller based on GenoGAM with performance that matches state-of-the-art methods. Moreover, GenoGAM provides significance testing for differential occupancy with controlled type I error rate and increased sensitivity over existing methods. By analyzing a set of DNA methylation data, we further demonstrate the potential of GenoGAM as a generic analysis tool for genome-wide assays.

We provide an R package called GenoGAM on Bioconductor [37] at `https://bioconductor.org/packages/release/bioc/html/GenoGAM.html`.

## 4.2 Background

This section briefly covers the experimental and methodical background necessary for this chapter. For a more detailed review of the material, the references cited herein cover these topics more thoroughly.

### 4.2.1 Next-generation sequencing

DNA sequencing is a method used to determine the specific sequence of nucleotides in a DNA molecule. DNA is comprised of a sequence of four nucleotides, adenine (A), cytosine (C), guanine (G), and thymine (T). Sequencing the DNA of organisms and individuals accurately and cheaply is of crucial interest to those working in fields such as synthetic biology and personalized medicine [87].

In 2003, the Human Genome Project published the first complete human reference genome [53]. Back then, the state-of-the-art technology was Sanger sequencing, which had already been in use for almost two decades [77]. It is a method that is slow and expensive compared to today's possibilities. The sequencing alone of this first human genome cost more than 500 million US dollars. As a comparison, the long-desired goal of reaching the "$1000 genome" (i.e., a fully sequenced human genome that costs no more than 1000 USD to produce) is very close to completion as of 2017. This enormous drop in cost was made possible by the development of next-generation sequencing (NGS) technology. NGS combines various techniques to speed up the preparation and sequencing so that massive numbers of fragments can be sequenced cheaply, accurately, and quickly [77, 68]. NGS also opens the door for large-scale comparative studies such as ChIP-Seq and RNA-Seq experiments.

NGS comprises many different sequencing techniques. The most important characteristics, however, are common to all NGS methods [13]. The DNA is split into millions of small fragments, typically in the range of 50–500 base pairs (bp, Figure 4.1). These fragments then have to be prepared by adding adapters to the DNA fragments and amplifying the templates [77]. The prepared templates can then be sequenced, usually by synthesizing the complementary DNA strand and imaging fluorescent signals that are emitted each time a new base is added. The fluorescence noise increases the longer a read gets, so the resulting *reads* are usually only between 30 and 50 base pairs (bp) long (Figure 4.1).

Two approaches are then available to sequence fragments. In *paired-end sequencing*, each fragment is sequenced from both sides. During the subsequent *mapping* (Section 4.2.3), we then align these reads to a reference genome, and find the positions of both reads and thus, implicitly, also know the unsequenced gap (Figure 4.1). This has the advantage that, after mapping, we have knowledge about the total insert length and thus the fragment midpoint. In *single-end sequencing*, only *Read 1* (Figure 4.1) is sequenced for each fragment.

The (average) fragment length then has to be estimated by some other method (Section 4.4.1).

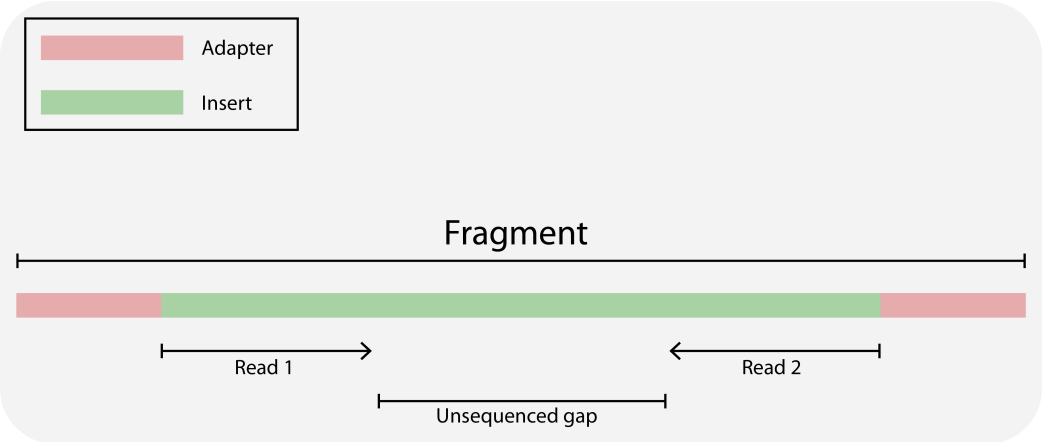



Figure 4.1: Schematic explanation of NGS fragments. In paired-end sequencing, the insert (green) is sequenced from both sides. In single-end sequencing, only *Read 1* is sequenced for each fragment.

### 4.2.2 ChIP-Seq

NGS can be applied to any library of DNA fragments. If one wants to sequence the unknown genome of a new species *de novo*, it is sufficient to take random DNA fragments from all over the genome [84]. However, it is often of interest to obtain only those fragments to which a specific protein is bound. This way, one can obtain genome-wide occupancy maps of where a chromatin-related protein binds and investigate how this occupancy map differs between experimental conditions or phenotypes.

To explore these protein-DNA interactions, chromatin immunoprecipitation followed by deep sequencing (ChIP-Seq) is the current reference method [91]. It is used to study a wide range of fundamental genome biology processes covering transcription, replication, and maintenance.

ChIP-Seq can be applied to study DNA-bound proteins of various functions and therefore with various patterns of distribution along the genome. These include transcription factors that bind to DNA sites and regulate (either promote or inhibit) the transcription from DNA to messenger RNA [55, 10], histone modifications [3, 10] which are found at nucleosomes, or the transcription [10] and replication machinery which are even more broadly distributed.

In this chapter, we analyze several common chromatin-related proteins of interest. First, DNA Polymerase II (PolII) is involved in DNA replication. Second, a histone modification, namely the trimethylation of the K4-Lysine of histone 3 (H3K4me3), is commonly found at

promoters of actively transcribed genes [83]. Third, the *Transcription factor II B* (TFIIB) contributes to forming the pre-initiation complex (PIC) and initiating the transcription of a gene [60].

The ChIP-Seq workflow is illustrated in Figure 4.2. Briefly, it proceeds as follows [25]: First, proteins that are currently bound to DNA are *cross-linked* so that they are not released in the subsequent steps. Then, during *fragmentation*, the DNA is sheared into short fragments between 200 bp and 1 kilobase (kb), by means of, for example, sound energy (a process called *sonication*). The *immunoprecipitation* step then uses protein-specific antibodies to extract only those DNA fragments to which a protein of interest is bound.

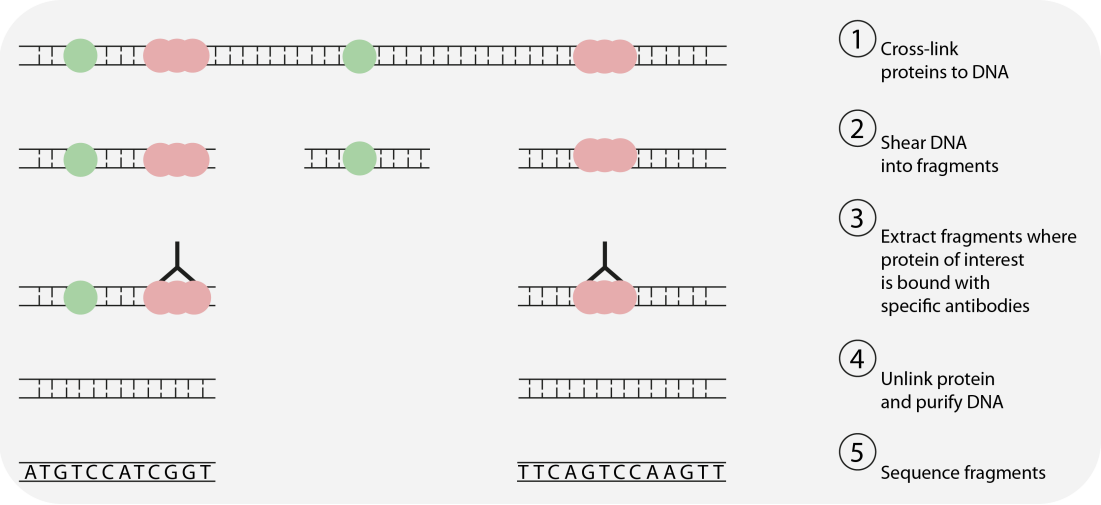



Figure 4.2: Schematic illustration of ChIP-Seq. In step 1, DNA-bound proteins are immobilized by cross-linking. The DNA is then sheared into short fragments between 200 and 1000 bp (step 2). Then, in step 3, antibodies that target the protein of interest extract only those fragments where one of these proteins is bound. Lastly, the DNA is purified (step 4) and fragments are sequenced (step 5).

The DNA fragments are then released, amplified, and sequenced (Section 4.2.1). Often, the quantities of interest are the occupancies relative to technical controls such as the Input (i.e., a sample that was not subject to the immunoprecipitation step), or differential occupancies between genetic backgrounds, treatments, or combinations thereof.

As an example, one could investigate the occupancy profile of PolII along a human genome. For this, an experiment would consist of one or multiple *replicates* of Input and IP, each. The Input sample is necessary to correct the IP sample for region-specific biases, for example because the amplification step is not exactly random but favors certain fragments based on their GC content [2]. An experiment should ideally consist of more than one replicate for each experimental condition in order to assess the reproducibility [63]. The number of fragments sequenced (the *sequencing depth*) is not equal across all replicates.

Therefore, the subsequent analysis steps have to consider this imbalance, especially when multiple replicates are available.

### 4.2.3 Bioinformatics

Bioinformatics is a relatively new, interdisciplinary field that combines knowledge from molecular biology, statistics, mathematics, physics, and computer science [9]. It arose as a response to the breakthroughs in sequencing technology, which made massive amounts of new data available to molecular biologists. To work with these large data sets, new methods and software had to be developed.

The analysis pipeline from a finished ChIP-Seq experiment to obtain an answer to a research question is long and complex. Bioinformaticians provide the necessary tools to cover these steps. For example, the first step is usually the *mapping* of the reads obtained from a NGS experiment to find its originating position in the reference genome. The presence of complicating factors such as sequencing errors and DNA differences between individuals or cells (e.g. deletions or insertions) make this task nontrivial. A widely-used mapping program is called Bowtie [66], which performs read alignment in a fast and memory-efficient manner.

The result of the mapping is usually a coverage track, which piles up the sequenced reads and thus shows the number of reads that overlapped a position as a function over the genomic position. Figure 4.3 shows how the mapped fragments build a coverage track.

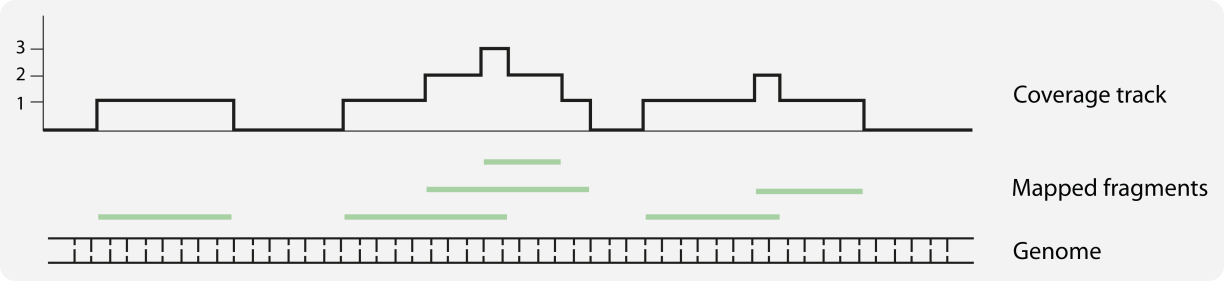



Figure 4.3: A coverage track is created by "piling up" the mapped fragments to create a function of read counts per base-pair

After mapping, a common subsequent analysis of ChIP-Seq data is *peak calling* [110], where practitioners aim to identify discrete points (peaks) in the genome where the coverage track shows a significant maximum (possibly after correction by an Input track).

Of course, the field of bioinformatics spans a vast number of other topics. This discussion should serve only as a brief explanation of the basics necessary for this chapter.

## 4.3 Introduction

Although ChIP-Seq is a generic methodology used to study protein-DNA interactions, statistical analysis methods have been so far dedicated to specific applications. Early work has focused on transcription factors with discrete binding sites, typically DNA motifs at promoters or transcriptional enhancers [55, 110]. ChIP-Seq read coverage then shows peaks localized at the binding sites. The aim of these statistical methods is to identify these peaks and their statistical significance, typically by controlling the false discovery rate. For example, MACS [110] is a widely used [32, 108] peak caller that assumes a Poisson distribution for the count data and computes peak significance based on a combination of global and local rate. ZINBA [89] combines a negative binomial mixture model for background and enriched regions with a zero-inflated component for regions with excessive zero counts. The specific calling of narrow and wide peaks was made possible by JAMM [51], which makes use of replicates, and is based on a mixture model of enriched and non-enriched regions.

A more sophisticated problem is to find regions with differential overall occupancies across conditions (i.e., a significant difference in the amount of DNA-bound protein in a region). This is done by testing for differences in number of reads overlapping the region [8]. Complementary to testing for overall occupancies, MMdiff [93] allows testing for differences in shapes in given regions. Lun et al. [70] provide a framework to test differential occupancies between conditions across windows in given regions while properly controlling for the false discovery rate. This approach allows testing for differences in overall occupancies and in shapes.

We have seen that practitioners rely on different statistical frameworks for peak calling tasks and differential occupancies. Furthermore, flexible handling of replicates and additional control factors is not always possible. Moreover, current methods rely on binning and/or sliding window techniques, whose choice of the window size is not data-driven but subjective. Another limitation is that the more general task of statistical inference of a genome-wide bias-corrected occupancy track is not addressed.

To resolve the aforementioned limitations, we developed Genome-wide Generalized Additive Models (GenoGAM), which provides a statistical framework to address all these issues. Our model describes the genome-wide occupancy of proteins by smooth functions, which facilitates downstream applications such as peak calling or differential binding analysis. GenoGAM normalizes for sequencing depth and can handle factorial experimental designs, including biological replicates and multiple controls. The amount of smoothing is estimated in an automatic, data-driven manner and thus avoids introducing subjectivity from the analyst. When analyzing differential binding in a factorial design, we obtain well-calibrated per-base-pair $p$-values. Application to data sets from human and yeast shows that GenoGAM is as performant as dedicated methods for peak calling and much more sensitive than state-of-the-art differential occupancy methods. By providing an approximation to a conventional generalized additive model (GAM [45]) that allows a data parallelism

implementation, GenoGAM scales linearly with the number of data points, making whole-genome applications computationally tractable. Our method provides a framework that is applicable not only to ChIP-Seq data but also to other next-generation sequencing data such as DNA methylation data (Figure 4.4a).

## 4.4 General methods

### 4.4.1 Preprocessing

Fragments were centered, reducing each fragment to a single data point. In case of single end data, the fragment length $d$ was 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. Then, the center was taken as the start of the read shifted by $\frac{d}{2}$ downstream.

### 4.4.2 Sequencing depth variations

Variations in sequencing depth were controlled by using size factors computed by DESeq2 [69] (version 2_1.10.0 here and after). This method robustly estimates fold-changes in overall sequencing depth by comparing read counts of predefined regions. The selection criteria for these regions were application-specific and are described in the respective sections.

For peak calling applications, the selected regions were the 1,000 tiles with the smallest $p$-value according to DESeq2 test for enrichment of IP over Input performed on total read counts per tile. This allowed to select tiles that were most likely containing peaks. For differential binding analyses, all tiles were considered.

### 4.4.3 A generalized additive model for ChIP-Seq data

We consider an experiment consisting of a set of ChIP-Seq samples. A data point $y_i$ is defined by the read coverage of a ChIP-Seq sample at a specific genomic position (Figure 4.3). We 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 fragment centers in sample $j_i$ centered at position $x_i$. For single-end libraries, the fragment center is estimated by shifting the read end position by a constant (Section 4.4.1). 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 and biases due to different fragment lengths. We model the counts $y_i$ using the following generalized additive model:

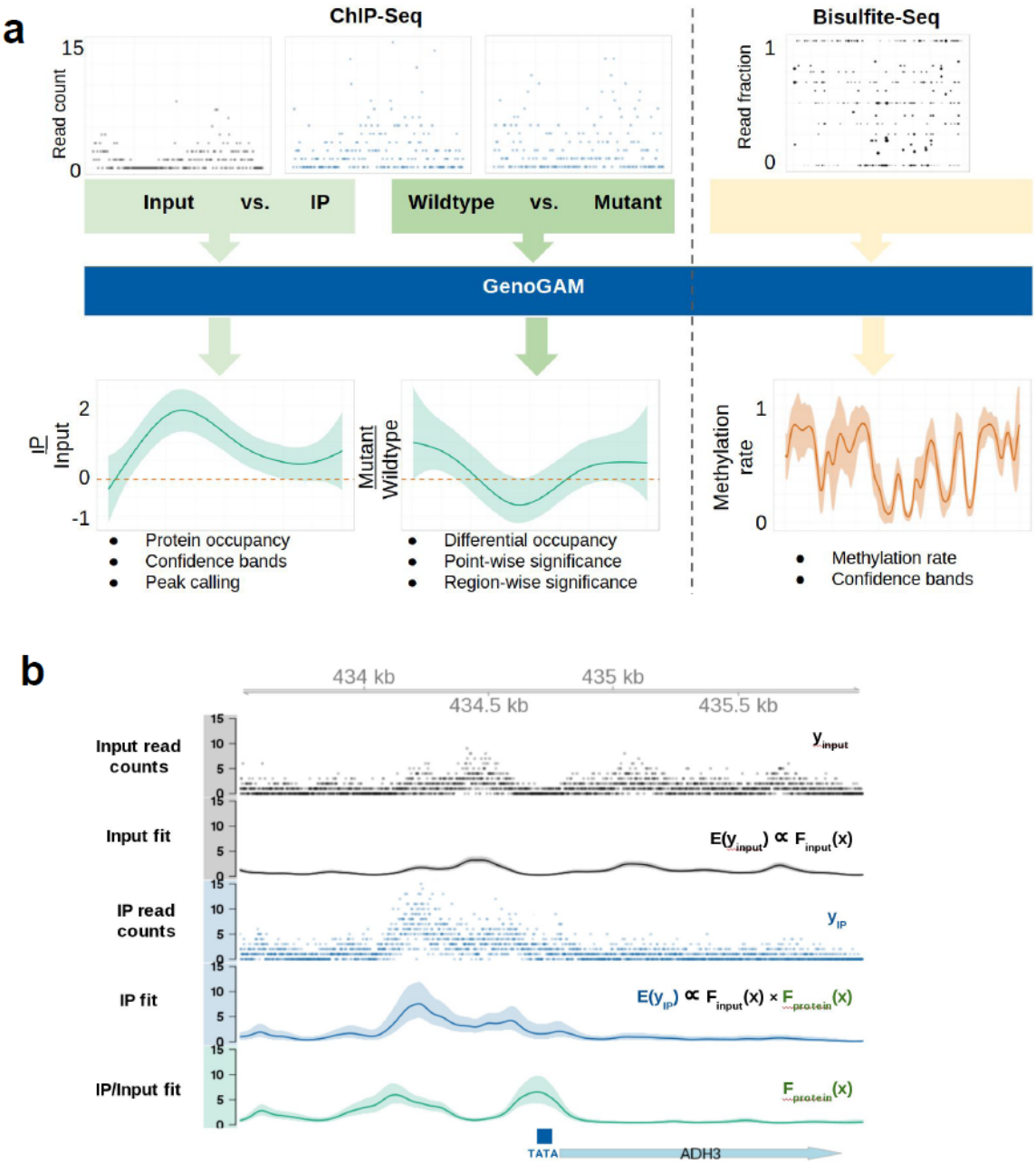



Figure 4.4: **GenoGAM concept and applications.** (**a**) GenoGAM provides a general framework to analyze ChIP-Seq data for both absolute (left arrow) and differential protein (center arrow) occupancy. It can also be applied to infer DNA methylation rate from bisulfite sequencing data (right arrow). (**b**) ChIP-Seq analysis with GenoGAM yields base-pair resolution occupancy profiles with confidence bands. Shown are Input (black) and IP (blue) centered read counts (dots) and fitted smooths (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 bottom. Simplified equations depict model constituents.

$$y_i \sim \mathrm{NB}(\mu_i, \theta) \tag{4.1}$$

$$\log(\mu_i) = o_i + \sum_{k=1}^{K} f_k(x_i)\, z_{j_i,k} \tag{4.2}$$

The counts $y_i$ are assumed to follow a negative binomial distribution with mean $\mu_i$ (Equation 4.1) and a dispersion parameter $\theta$ that relates the variance to the mean such that $\mathrm{Var}(y_i) = \mu_i + \mu_i^2/\theta$. In this way, the model can account for over-dispersion [8]. The logarithm of the mean $\mu_i$ is the sum of an offset $o_i$ and one or more smooth functions $f_k$ (Equation 4.2). The offsets $o_i$ are predefined sample-specific constants that account for sequencing depth variations (Section 4.4.2). The indicator variable $z_{j_i,k}$ equals 1 if the smooth function $f_k$ contributes to the mean counts of sample $j_i$ and 0 otherwise. As demonstrated in the following analyses, this flexible formulation allows modeling IP versus Input experiments as well as factorial experimental designs.

We modeled IP versus Input experiments using GenoGAM with two smooth functions. First, $f_{\text{input}}$ contributes to both Input and IP samples. More specifically, $f_{\text{input}}$ models local ChIP-Seq biases common to Input and IP. Second, $f_{\text{protein}}$ contributes only to IP samples and models the protein log-occupancy up to one genome-wide scaling factor. Figure 4.4b shows the application of this model to one ChIP-Seq library for the *S. cerevisiae* general transcription factor TFIIB and its Input control (Section 4.5).

In GenoGAM, the smooth functions are represented by cubic splines, which are written as linear combinations of a set of regularly spaced B-spline basis functions $b_r$, i.e. $f_k(x) = \sum_r \beta_r b_r(x)$. We chose second order B-splines as basis functions, which are bell-shaped cubic polynomials over a finite support [26]. To avoid overfitting, we use P-splines (penalized B-splines [30]), which carry out additional smoothing of the functions $f_k$ out by penalizing the second order differences of the spline coefficients, which approximately penalizes second order derivatives of $f_k$. The optimization criterion for P-splines is the sum of the negative binomial log-likelihood (depending on the response vector $y$ and the vector $\boldsymbol{\beta}$ containing the coefficients of all smooth functions) plus a penalty function that is weighted by the smoothing parameter $\lambda$:

$$\hat{\boldsymbol{\beta}} = \mathrm{argmax}\ \ l_{\mathrm{NB}}(\boldsymbol{\beta}; \mathbf{y}, \theta) - \lambda \boldsymbol{\beta}^\top \mathbf{S} \boldsymbol{\beta}, \tag{4.3}$$

where S is a symmetric positive matrix that encodes the squared second-order differences of the coefficients $\boldsymbol{\beta}$ [30]. Large values of $\lambda$ yield smoother functions. This regularization allows dense placements of the basis functions (between 20 and 50 bp), while relying on the smoothing parameter $\lambda$ to protect against overfitting. A single smoothing parameter common to all smooth functions proved to be sufficient for our applications. For given $\lambda$

and $\theta$, the model fitting was performed using penalized iteratively re-weighted least squares [106].

The penalized likelihood can also be interpreted in a Bayesian fashion [106], where a multivariate Gaussian prior is placed on the coefficient vector $\boldsymbol{\beta}$. Large-sample approximations then yield a multivariate Gaussian posterior distribution 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 [106]. An example of the fitted smooth functions and their confidence bands for the yeast transcription factor TFIIB is shown in Figure 4.4b.

### 4.4.4 Genome-wide estimation of smoothing and dispersion parameters

To determine the optimal values for $\lambda$ and $\theta$, generalized cross-validation, the default approach for GAMs which is based on an analytical leave-one-out large-sample approximation [106], yielded excessively fluctuating fits indicative of overfitting. We therefore developed an empirical cross-validation scheme. To reduce computational time, cross-validation was performed on a subset of all data. For this purpose, we selected a sufficiently large set of distinct regions that are long enough to not suffer from border effects common to spline fitting. Using 40 or more distinct regions containing at least 60 basis functions gave satisfactory empirical results (Table 4.1). Also, it was important to select regions relevant for the desired application. For peak calling purposes, regions were selected that had the most significant fold change of IP versus Input read counts.

The global parameters $\lambda$ and $\theta$ were then estimated once on this subset of all tiles. The selection of relevant tiles for cross-validation was application-specific as outlined in the respective sections below. In each region, a 10-fold cross-validation was performed, where a tenth of the data points were randomly removed, the model was fitted on the remaining data points, and the log-likelihood of the left-out data points was computed.

Short range auto-correlations are strong in ChIP-Seq data and are not fully controlled by replicates or Input experiments. To avoid overfitting due to short range autocorrelation, each cross-validation fold did not consist of randomly selected single base pairs but of short intervals. When replicate samples were available (for all except the TFIIB data set), intervals could be of greater length because the model can predict samples from the respective replicates. We set it to twice the estimated fragment length. In the absence of replicates (TFIIB data set), interval length was set to 20 bp (i.e., approximately a tenth of the average fragment length.).

For a given pair of values for $\lambda$ and $\theta$, the score function was 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 $\theta$ and $\lambda$ showed that the out-of-sample log-likelihood was typically unimodal. We therefore used a Nelder-Mead optimization [82] (R function `optim`) to jointly fit the two parameters in a computationally faster way than a grid search.

### 4.4.5 Fitting a GAM genome-wide

Since the computation time of a GAM grows polynomially with the number of basis functions, fitting one model to a whole chromosome is unfeasible. Instead, we propose to fit separate GAMs on sequential, overlapping intervals (or *tiles*, Figure 4.5a). We partitioned each chromosome into equally-sized intervals called *chunks*. Tiles were then defined as chunks extended on both sides by equally-sized *overhangs*. The generalized additive model was fitted on each tile separately using the `gam` function of the R package `mgcv`. Point estimates at each base pair of the smooth functions and their standard errors were extracted with the `predict` function on the fitted object setting the `type` parameter to `iterms`. The tile fits were then restricted to their chunk to compose the chromosome-wide fit.

As the length of the overlapping intervals increases, agreement of the fit at the midpoint of the overlap increases. A genome-wide fit is obtained by joining together tile fits at chunk ends (Figure 4.5a). This approximation yields computation times that are linear in the number of basis functions at no practical cost in precision (Figure 4.5b). Furthermore, it allows for parallelization, with speed-ups being linear in the number of cores (Figure 4.5c). This approximation parallelizes the computation over the data, which will allow future implementation of GenoGAM in map-reduce frameworks such as Spark [109].

### 4.4.6 Peak calling

#### 4.4.6.1 GenoGAM-based peak calling and $z$-score

Because analytical derivatives of P-splines are available, identifying peaks of the protein occupancy $f_{\text{protein}}$ is straightforward by extracting local maxima where $f'_{\text{protein}}(x) = 0$ and $f''_{\text{protein}}(x) < 0$ (Figure 4.6).

Values of first and second derivatives of fitted smooth functions were obtained by multiplying the estimated coefficients with the corresponding derivatives of the B-splines as obtained from the `spline.des` function of the R package `splineDesign`. Local extrema (at base pair resolution) were identified as positions at which the sign of the first derivative differed from the one of the preceding position.

To assess the statistical significance of peak heights, we introduced an empirical $z$-score that contrasts the estimate of the log-occupancy $\mu$ at the peak to a robust estimate of the background log-occupancy level $\mu_0$, taking both background level variance $\sigma_0^2$ and uncertainty of peak height $\sigma^2$ into account:

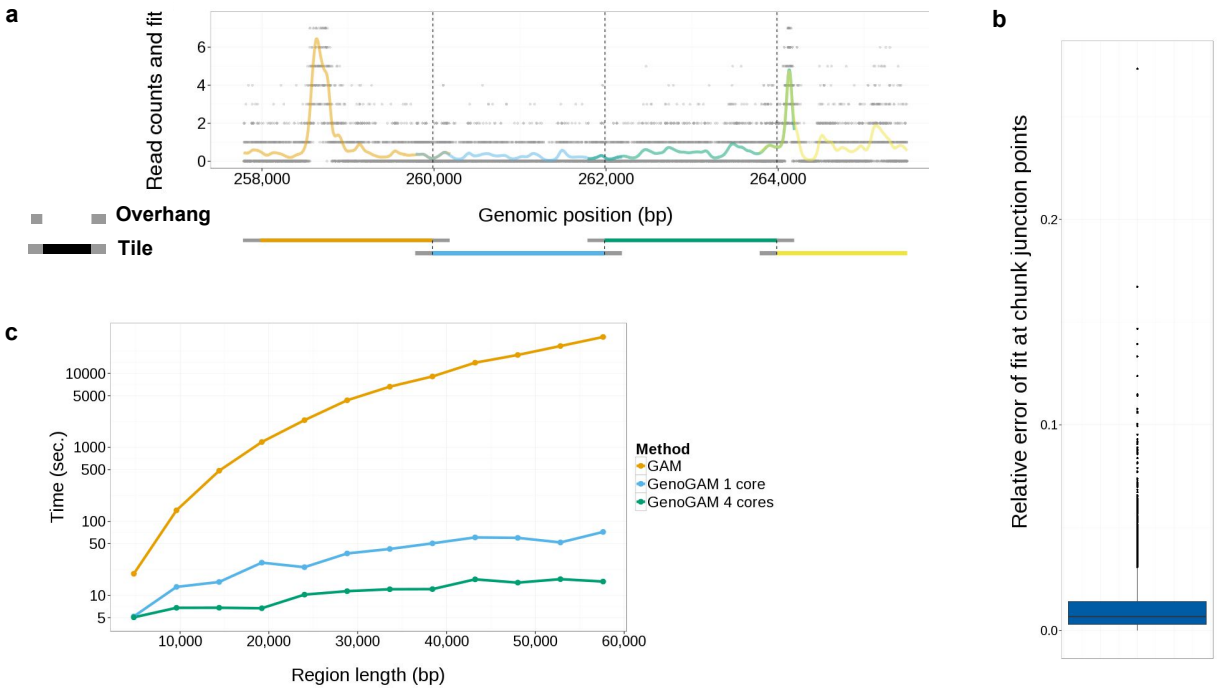



Figure 4.5: **Per tile-parallelization allows map-reduce implementation of GAM.** (**a**) Read count (black dots, capped at $\geq 7$) and predicted rates (orange, blue, green, and yellow transparent lines) for four successive tiles (lower track). Vertical dashed lines denote the junction points. (**b**) Distribution of the relative error (difference over mean) at the junction point of two neighboring tiles, for an overhang of 8 basis functions. (**c**) 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.

$$z = \frac{\mu - \mu_0}{\sqrt{\sigma^2 + \sigma_0^2}}. \tag{4.4}$$

For the $z$-score, we estimated $\mu_0$, the global background mean, and $\sigma_0^2$, the global background variance of $f(x)$ as follows: In order to account only for the background without potential peaks, $\mu_0$ was estimated as the `shorth` (from the Bioconductor package `genefilter`) of all fitted values $f(x_i)$, $i = 1, \ldots, n$, which is defined as the midpoint of the shortest interval containing half of the data. The fitted values smaller than the shorth were mirrored on it, such that a symmetric density was created that excludes the values larger than the shorth, in particular those high values representing peaks. The variance of this newly created distribution was then estimated in a robust fashion by the *median absolute deviation* (MAD) giving $\sigma_0^2$ (Figure 4.6).

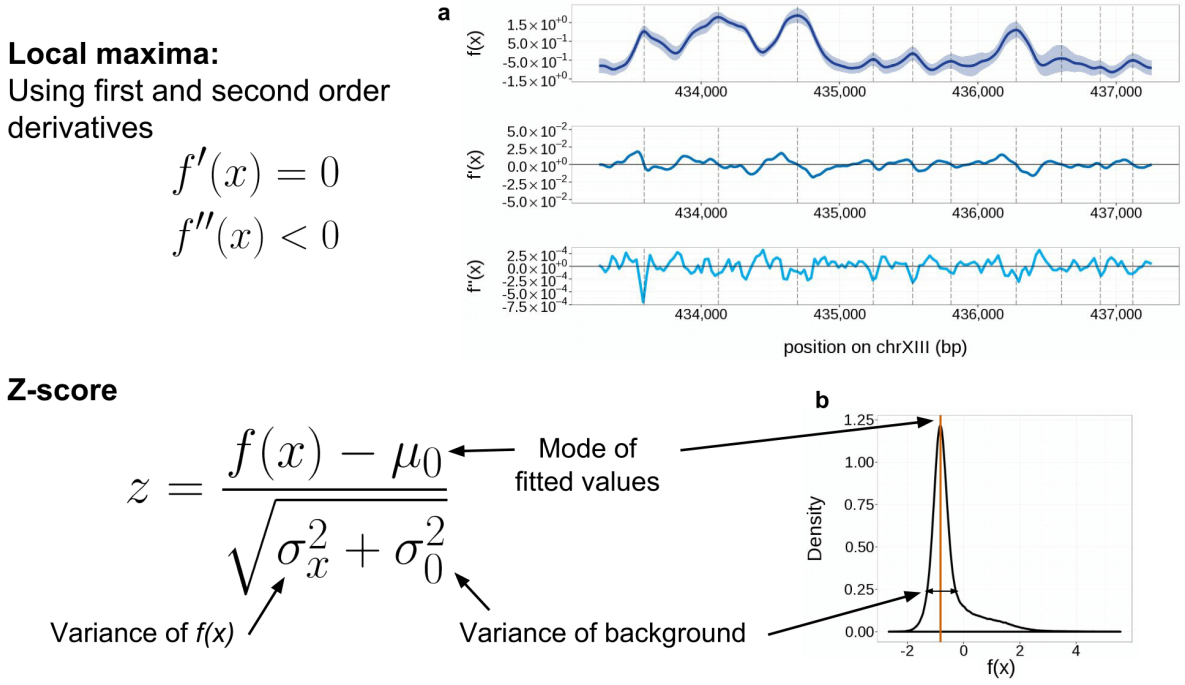



Figure 4.6: **Peak calling.** (**a**) Fit of protein log-occupancy $f_{\text{protein}}(x)$ (top), its first derivative $f'_{\text{protein}}(x)$ (middle) and its second derivative $f''_{\text{protein}}(x)$ (bottom). Shown is the same data as in Figure 4.4b, but extended a little further. (**b**) Illustration of the $z$-score computing procedure.

#### 4.4.6.2 False discovery rate for peaks

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 inverting the role of Input and IP [110]. We therefore estimated the false discovery rate (FDR) using the $z$-score distributions of the local minima.

We therefore performed a peak-calling on the negative estimated function, i.e. $-f_{\text{protein}}$. Their $z$-scores were obtained by recomputing $\mu_0$ and $\sigma_0$ and applying the same formula. The FDR for a given minimum $z$-score $z$ was estimated by $\frac{|V_z|}{|P_z|+|V_z|}$, where $P_z$ and $V_z$ are the sets of peaks and valleys, respectively, with a $z$-score greater than or equal to $z$.

#### 4.4.6.3 Parameters of the competitor methods, MACS, JAMM, and ZINBA

Version 2 of the MACS software, MACS2 [110], 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 [51] 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, we used the complete, sorted (by score) peak list and took the top $N$ results 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 we used the same number of peaks that MACS reported.

For ZINBA [89], 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 to `method = "mixture"`.

## 4.5 Yeast TFIIB data set

This data set came from the yeast organism and was a ChIP-Seq experiment done on the TFIIB transcription factor. This transcription factor often shows peaks around the well-characterized DNA element *TATA-box*. The data set contained two samples, one Input

and one IP sample, both without replicates. The experimental details are available in Appendix B.1.

## 4.5.1 Methods

### 4.5.1.1 GenoGAM model

Because the data set contained no replicates, there was no need for an offset. We used the following GenoGAM model:

$$y_i \sim \mathrm{NB}(\mu_i, \theta)$$
$$\log(\mu_i) = f_{\mathrm{input}}(x_i) + f_{\mathrm{protein}}(x_i) z_{j_i,\mathrm{protein}},$$

where $z_{j_i,\mathrm{protein}} = 1$ whenever $j_i$ is the index of an IP sample and $z_{j_i,\mathrm{protein}} = 0$ whenever $j_i$ is the index of an Input sample. Further parameter details are given in Table 4.1.

| | **TFIIB** | **ENCODE TFs** | **Differential binding** | **DNA methylation** |
|---|---|---|---|---|
| Genome | SacCer2 | hg19 | SacCer3 | mouse build NCBI37 |
| Single/paired end | single end | single end | paired end | paired end |
| Knot spacing (bp) | 25 | 25 | 50 | 2000 |
| Tile size (knots) | 100 | 100 | 60 | 60 |
| Overhang size (knots) | 8 | 8 | 15 | n/a |
| Tiles for size factors | n/a | 1000 | all | n/a |
| Tiles for CV | 40 | 40 | 50 | n/a |
| CV intervals | 20 | 400 | 400 | n/a |

Table 4.1: **Parameter settings.** The table shows all parameter settings that were used to perform the analyses. The data sets are indicated in bold in the first row. As analysis on DNA methylation data was only performed as a proof of concept on one tile, some options are not applicable.

### 4.5.1.2 TATA-box mapping

For about 20% of yeast promoters, recruitment of TFIIB is triggered by the TATA-box, providing at these promoters a ground truth for a TFIIB occupancy peak [11]. We mapped 1,105 TATA-boxes genome-wide by regular expression of a consensus motif and considered 1 kb regions centered on TATA-boxes for benchmarking.

Promoter TATA-boxes where defined as instances of the motif TATAWAWR [11] at most 200 bp in 5'-direction and 50 bp in 3'-direction of one of the 7,272 transcript 5'-ends reported by Xu et al. [107].

### 4.5.2 GenoGAM provides a competitive peak caller

We first compared the peak-calling performance of GenoGAM, MACS [110], JAMM [51] and ZINBA [89] in identifying binding sites of TFIIB.

In the regions defined above, significant peaks (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, Figure 4.7a).

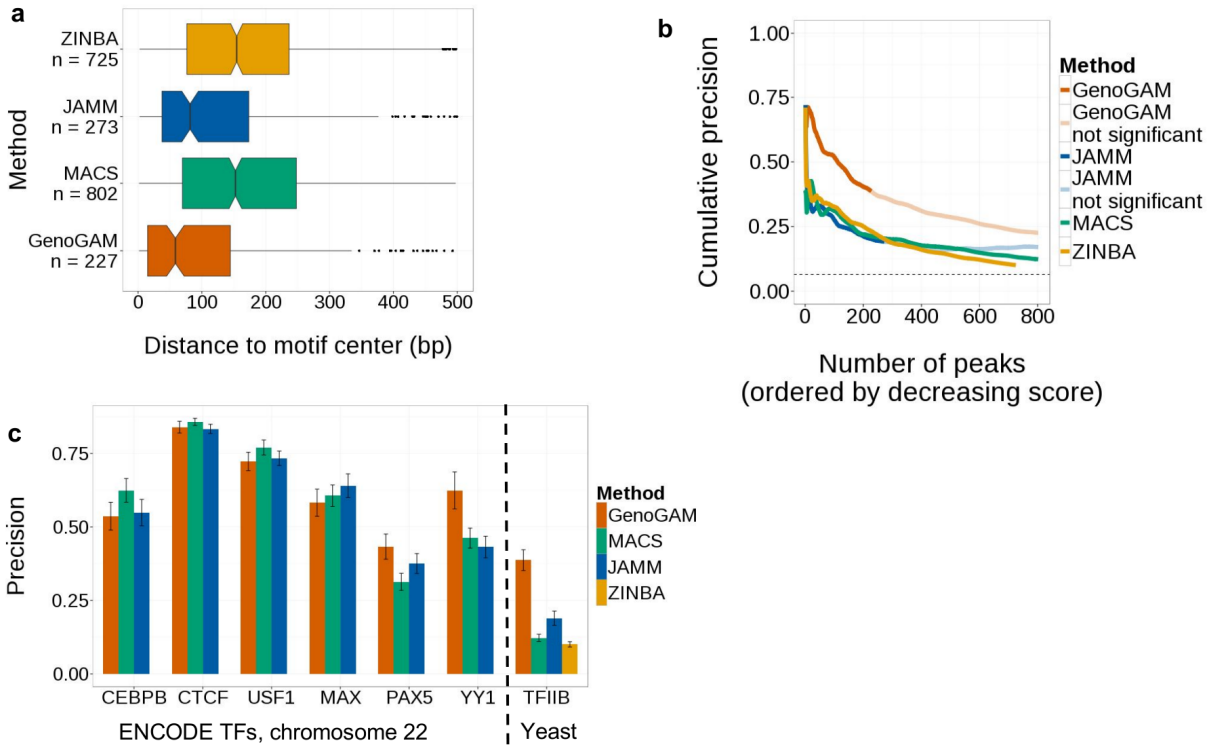



Figure 4.7: **GenoGAM identifies protein binding sites with similar accuracy to state-of-the-art peak callers.** (**a**) Boxplots of distances between significant peaks and TATA-box (FDR $< 0.1$, Section 4.4.6.2) for the yeast TFIIB data set (Section 4.5) for GenoGAM (orange), MACS (green) and JAMM (blue) 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). (**c**) Proportion of significant peaks called that are 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 data set.

Moreover, the proportion of peaks within 30 bp of a TATA-box center was twice as high as for any other method independent of the number of reported peaks (Figure 4.7b), showing

that the improvement was robust to the score threshold.

We next investigated the reason for the drastic differences observed in the yeast TFIIB data set between GenoGAM and the other methods. The TATA-box region upstream of the *IDH2* gene illustrates the issue (Figure 4.8a). 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 [110, 51]. Statistical significance increases with both effect size and sample size, leading to peak calls biased toward positions with high total counts in IP and Input (Figure 4.8a). 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 (Figure 4.8b), generalizing the observations made for *IDH2*. The yeast TFIIB data set was sequenced at a much higher coverage than the ENCODE data set (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, we anticipate that robust fold-change estimates as provided by GenoGAM will be a more sustainable criterion than mere significance for calling peak positions.

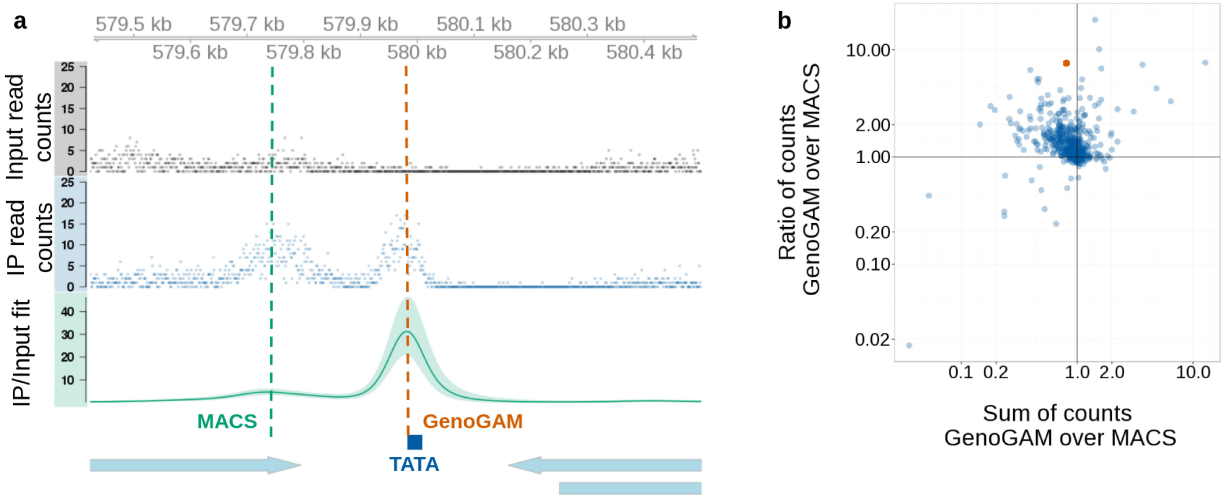



Figure 4.8: **Peak position represent maximal fold change rather than maximal significance.** (**a**) Example region in yeast with Input (black dots), IP (blue dots) and the smooth function IP over Input with 95% confidence interval (green line) showing a correctly identified peak position by GenoGAM (orange vertical dashed line) and an incorrect position by MACS (green vertical dashed line), due to enrichment in the Input sample. (**b**) Scatterplot of the sum of counts (Input + IP) versus 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)

## 4.6 ENCODE transcription factors

Next, we performed a similar peak-calling benchmark than in Section 4.5.2 on the human chromosome 22 for 6 transcription factors of the ENCODE project [32] selected to be representative of accuracies in predicting ChIP-Seq peak positions from sequence motifs [5] (CEBPB, CTCF, MAX, USF1, PAX5, and YY1).

### 4.6.1 Methods

#### 4.6.1.1 Data processing

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 data sets contained two biological replicates for the protein samples and at least one Input sample. However, the library sizes of the Input samples were so small that including them resulted in higher uncertainty about the peaks, for both our approach and alternative approaches. We therefore conducted the analyses without correction for Input.

#### 4.6.1.2 GenoGAM model

The data set was modeled separately for each transcription factor. Each data set consisted of IPs with replicates. The following GenoGAM model was used:

$$\begin{aligned} y_i &\sim \mathrm{NB}(\mu_i, \theta) \\ \log(\mu_i) &= \log(s_{j_i}) + f_{\mathrm{protein}}(x_i), \end{aligned}$$

where the offsets $\log(s_{j_i})$ are log-size factors computed to control for sequencing depth variation between the replicates (see Section 4.4.2). Further parameter details are given in Table 4.1.

#### 4.6.1.3 Transcription factor motif mapping

Motif occurrences in the genome were determined by FIMO [42] using the default threshold $10^{-4}$ with position weight matrices (PWMs) from the JASPAR 2014 database [74] with the following IDs: CEBPB: MA0466.1, CTCF: MA0139.1, MAX: MA0058.1, PAX5: MA0014.2, USF1: MA0093.2, YY1: MA0095.2

### 4.6.2 Results

On these data, GenoGAM performance was comparable to the other methods (95% bootstrap confidence intervals, Figure 4.7c for significant peaks, Figure 4.9 for distance distributions, and Figure 4.10 for all cutoffs). Hence, although GenoGAM is a general framework for ChIP-Seq analysis, it nonetheless provides a peak caller that is at least as performant as dedicated tools.

Two problems arose when analyzing ChIP-Seq data on the human organism. First, genomes with more than $2^{31}$ base pairs ($\approx$ 2.147 billion), e.g., the human genome, have to be analyzed one chromosome at a time. Additionally, the running time for the human genome was still rather long, taking between one and two days on a 60-core cluster, even when the data is pre-filtered for enriched regions.

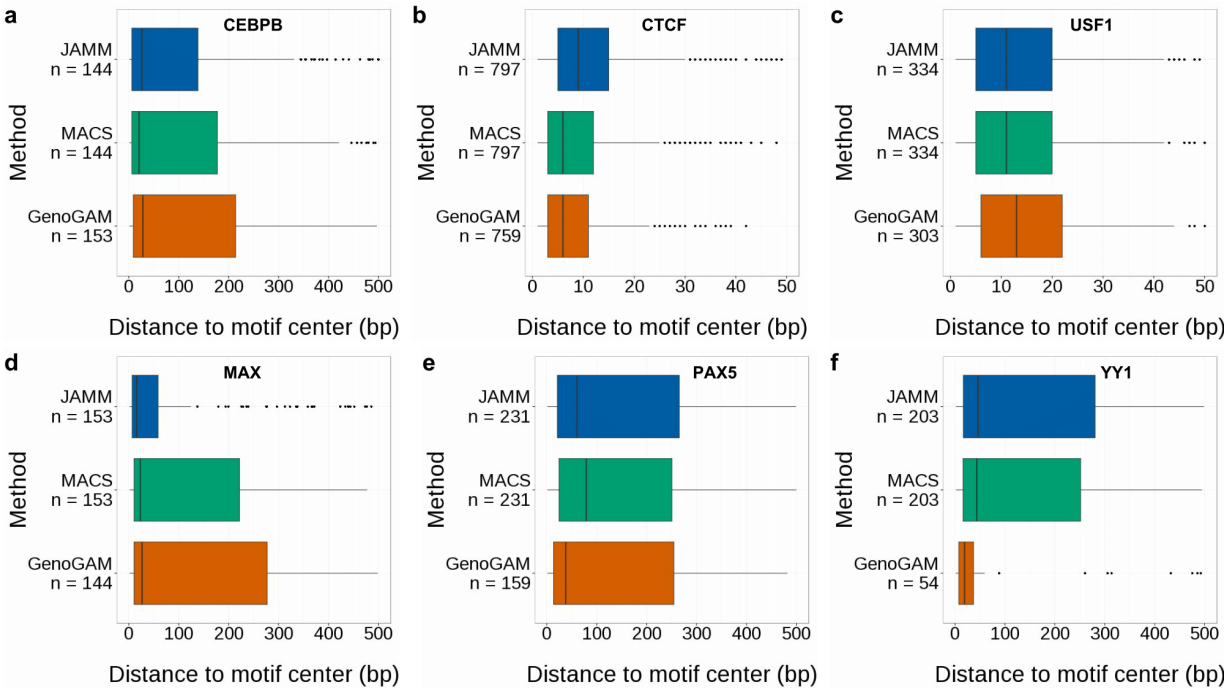



Figure 4.9: **Distances to motif center for ENCODE data.** As in Figure 4.7a for the ENCODE transcription factors CEBPP (**a**), CTCF(**b**), USF1 (**c**), MAX (**d**), PAX5 (**e**), and YY1 (**f**).

## 4.7 Differential binding

To assess the performance of GenoGAM for calling differential occupancies, we re-analyzed the histone H3 Lysine 4 trimethylation (H3K4me3) ChIP-Seq data of a study [99] comparing wild type yeast versus a mutant with a truncated form of Set1, the H3 Lysine 4

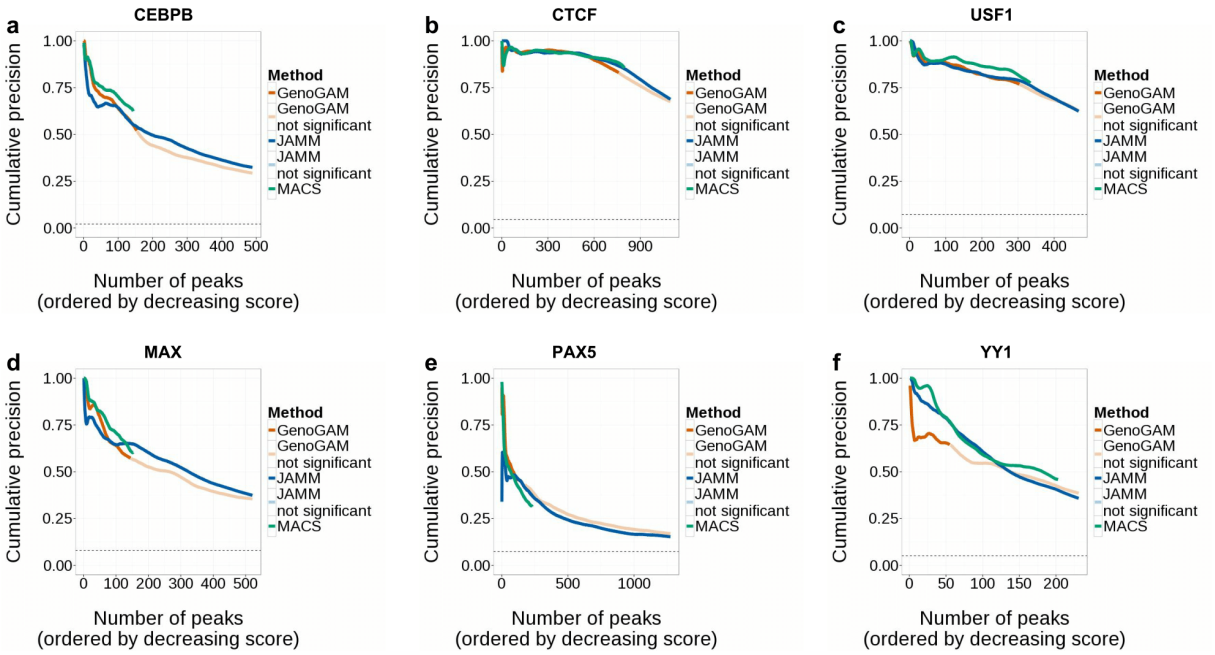



Figure 4.10: **Proportion of peaks for ENCODE data.** As in Figure 4.7b for the ENCODE transcription factors CEBPP (**a**), CTCF(**b**), USF1 (**c**), MAX (**d**), PAX5 (**e**), and YY1 (**f**).

methylase. H3K4me3 is a hallmark of promoters of actively transcribed genes. Thornton et al. [99] have reported genome-wide redistribution of H3K4me3 in the truncated Set1 mutant, which is depleted at the promoter and enriched in the gene body. Details on the data processing are available in Appendix B.2.

## 4.7.1 Methods

### 4.7.1.1 GenoGAM model

This data set consisted of four samples: two biological replicate IPs for the wild type strain, and two biological replicate IPs for the mutant strain. We used the following GenoGAM model:

$$\begin{aligned} y_i &\sim \mathrm{NB}(\mu_i, \theta) \\ \log(\mu_i) &= \log(s_{j_i}) + f_{\mathrm{WT}}(x_i) + f_{\mathrm{mutant/WT}}(x_i) z_{j_i,\mathrm{mutant}} \end{aligned}$$

where $z_{j_i,\mathrm{mutant}} = 1$ for the two mutant samples and 0 for the 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 (Section 4.4.2). Further parameter details are given in Table 4.1.

#### 4.7.1.2 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 were tested assuming approximate normal distribution of the corresponding $z$-score, i.e.:

$$T_k(x) \sim N(0, 1)$$

$$T_k(x) = \frac{\hat{f}_k(x)}{\hat{\sigma}^2_{f_k(x)}}$$

where $\hat{f}_k(x)$ and $\hat{\sigma}^2_{f_k(x)}$ denote point estimate and standard error of the smoothed value [106] as returned by the function `predict(..., type="iterms", se.fit=TRUE)` of the R package `mgcv`.

#### 4.7.1.3 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). Alternatively, non-contiguous positions, e.g. all exons of a gene, could also make up a single region. Regions can be defined a priori or defined on the data using independent filtering [19]. It is crucial to choose a filtering method that is independent of the test statistic of the subsequent test under the null hypothesis, since otherwise the false positive rate will be larger than reported [19]. 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 [70].

For $j \in 1, .., p$, we then defined $H_0^j$ as the composite null hypothesis that the smooth function $f_k$ values 0 at every position of the region $R_j$:

$$H_0^j = \bigwedge_{x_i \in R_j} (f_k(x_i) = 0)$$

We then obtained $p$-values for each region while controlling the false discovery rate with the following procedure [70]:

1. Position-level $p$-values at all region positions were computed using position-level significant testing as described above.
2. Within each region $R_j$, position-level $p$-values were corrected for multiple testing using the Hochberg family-wise error rate (FWER) correction [48]. The Hochberg correction was applied because it has a higher power than Bonferroni's or Holm's method, and the prerequisite that individual test statistics can not be negatively dependent [17] is given because position-level $p$-values of one smooth function are positively associated. The $p$-value for the null hypothesis $H_0^j$ was then computed as the minimal Hochberg-corrected position-level $p$-value. This step gives one $p$-value per region.
3. To then control the FDR across all regions, we used the Benjamini-Hochberg procedure [16] applied to the region-level $p$-values from step 2.

#### 4.7.1.4 Benchmarking

The R/Bioconductor packages `DESeq2` [69], `MMDiff` [93], and `csaw` [70] were applied on original count data and on the base-level permuted data set, for all genes. The log-size factors were set to 0 for all methods when applied to the permuted data sets. DESeq2 was applied with default parameters. MMDiff was applied with a bin length of 50 bp, the DESeq method for the normalization factor, and the Maximum Mean Discrepancy (MMD) histogram distance. The csaw method was applied with a window size of 150 bp and otherwise default parameters. The window size was determined by a grid search (see Figure 4.11c), choosing the window size with the most significant genes. In particular, csaw uses a different procedure to estimate normalization factors than DESeq and MMDiff. We used the default one as it was in favor of csaw for returning more significant genes.

### 4.7.2 Higher sensitivity in testing for differential occupancy

We modeled the data with GenoGAM using one smooth function $f_{\text{WT}}$ for the wild type reference occupancy, and a second smooth function $f_{\text{mutant/WT}}$ for the differential effect. The offsets were computed to control for variations in sequencing depth between replicates and overall genome-wide H3K4me3 level (Section 4.4.2). This yielded base-level log-ratio estimates and their genome-wide 95% confidence bands (Section 4.7). Figure 4.11a shows the data and the resulting fit at the gene *YNL176C*, consistent with the report of reduced binding at promoter regions.

As mentioned above, the confidence bands are Bayesian credible intervals. Previous studies based on simulated data showed that these confidence bands have close to nominal coverage probabilities and can be used instead of frequentist confidence intervals [73]. We estimated base-level $p$-values using the point-wise estimates and standard deviations (Section 4.7.1.2).

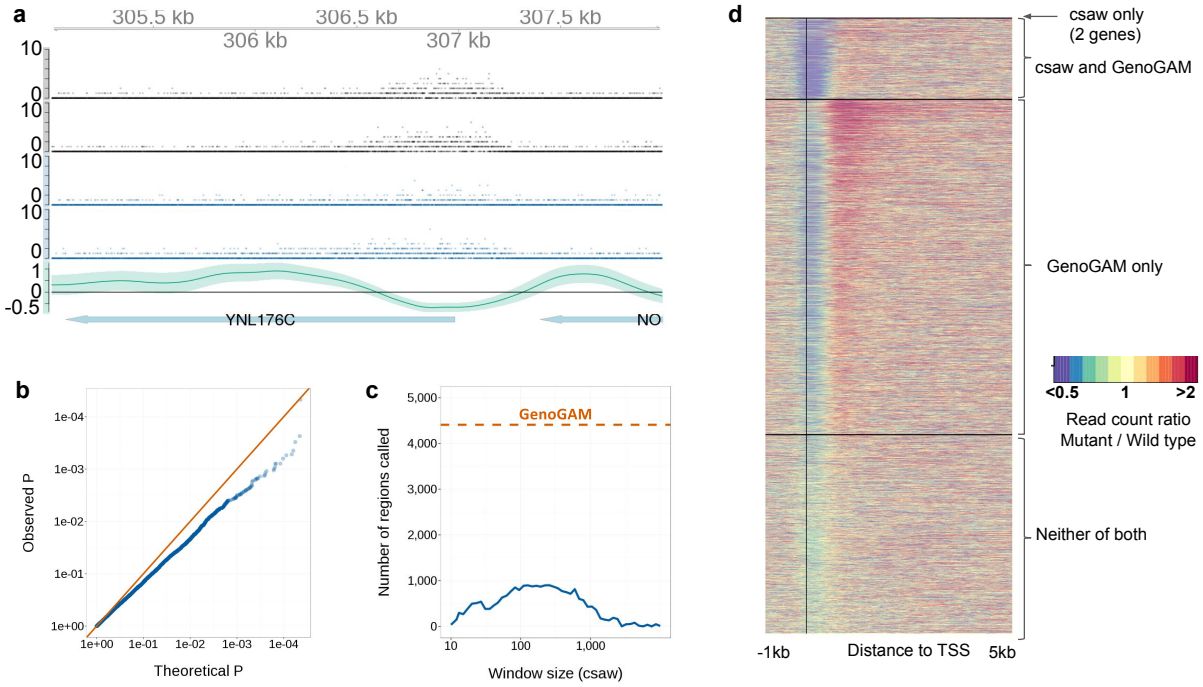



Figure 4.11: **Statistical testing for factorial designs.** (**a**) Read counts (dots) and fitted rates with 95% confidence bands for wild-type (black) and mutant (blue) and the log-ratio of mutant over wild-type with confidence band (bottom row, green) around *YNL176C*. (**b**) Empirical ($y$-axis) versus theoretical ($x$-axis) $p$-values in base-level permuted count data (Section 4.7.1.3). Shown are $p$-values at every 200 bp. (**c**) Number of genes with significant differential occupancies in mutant over wild type (FDR $< 0.1$) reported by GenoGAM (orange) and by csaw (blue) as function of window size ($x$-axis). (**d**) 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 as significant. From top to bottom: csaw only (2 genes), csaw and GenoGAM (861 genes), GenoGAM only (3,548 genes) and none (2,196 genes). Within each group, genes are ordered by $p$-values (lowest to highest from top to bottom). The "csaw and GenoGAM" group is sorted by GenoGAM $p$-values.

To empirically verify that the $p$-values were at least conservative, we created a negative control data set 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 [73]. Nonetheless, per-base-pair $p$-values in this negative control experiment were slightly overestimated (i.e., conservative; see Figure 4.11b). 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 be done using either the Benjamini-Hochberg procedure [16] or more sophisticated procedures that exploit dependencies between adjacent positions [103].

Complementary to de novo identification, predefined regions, such as genes, can be tested for differential occupancies. To test for differences at any position in a region using GenoGAM, we propose to apply Hochberg's procedure to correct the point-wise $p$-values for multiple testing, and to report the smallest of these corrected $p$-values (Section 4.7.1.3). To validate this approach, we compared GenoGAM against the following three approaches: csaw [70], which also tests for differences at any position in the regions, DESeq [8], which tests for differences in the overall occupancies, and MMdiff [93], which tests for differences of distribution within the regions but not overall occupancy (Section 4.7.1.4). All investigated methods empirically controlled the type I error on the permuted data set at a 5% nominal level (Figure 4.12). On the original data set, the least number of significant genes (FDR $< 0.1$) were identified by DEseq (735) and MMdiff (5). The csaw algorithm gave up to 863 significant genes but the number of identified genes depended strongly on the choice of the window size (Figure 4.11c). Of all methods, GenoGAM was the most sensitive, reporting 4,409 significant genes. Of the 863 significant genes found by csaw, 861 genes were found by GenoGAM as well, indicating that GenoGAM captured the same signal but with a higher sensitivity (Figure 4.11d). 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 (Figure 4.11d), indicating that GenoGAM captured true biological signal.

## 4.8 Methylation data

Generalized additive models allow any distribution of the exponential family for the response. Therefore, GenoGAM can also be used to model proportions using the binomial distribution. For ChIP-Seq data, a log-linear predictor-response relationship of the form in Equation 4.2 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 [102].

To test the flexibility of GenoGAM, we conducted a proof-of-principle study on modeling

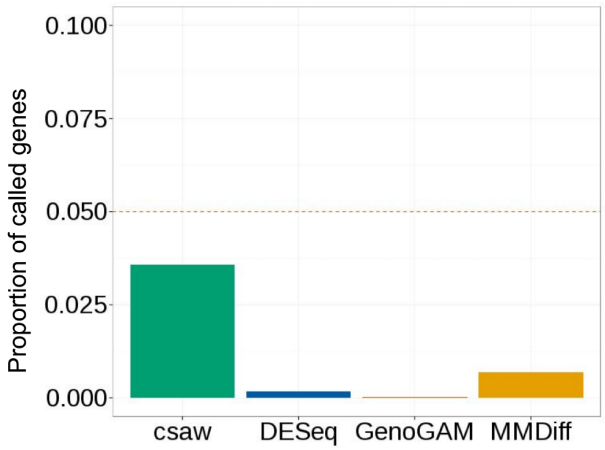



Figure 4.12: **Significant genes on permuted data.** Proportion number of called genes on permuted data (false positives) at nominal $p$-value 0.05 (red dashed line).

bisulfite sequencing of bulk embryonic mouse stem cells grown in serum [95]. Bisulfite sequencing quantifies the methylation rate by converting cytosine residues to uracil, but leaving 5-methylcytosine residues unaffected. At each cytosine, the data consisted of the total number $n_i$ of fragments overlapping the cytosine and the number $y_i$ of these fragments with a methylated cytosine. 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.

## 4.8.1 Methods

### 4.8.1.1 Data processing

We obtained the data in text table format from Smallwood et al. [95] from the Gene Expression Omnibus (GEO) repository `http://www.ncbi.nlm.nih.gov/gds`. The data provided, was one record per CpG site, with the number of methylated and unmethylated fragments at the respective site. We used a Python script to process this data into bins of 3,000 bp width every 600 bp, as was done in the original paper.

#### 4.8.1.2 GenoGAM model

To model $y_i$, the number of reads of methylated state, out of $n_i$, the total number of reads, we used the quasi-binomial model defined by:

$$\mathrm{E}(y_i/n_i) = \mu_i \tag{4.5}$$

$$\log(\frac{\mu_i}{1-\mu_i}) = f_{\text{methylation}}(x_i) \tag{4.6}$$

$$\mathrm{Var}(y_i/n_i) = \theta \cdot \frac{\mu_i(1-\mu_i)}{n_i}, \tag{4.7}$$

where the scale parameter $\theta > 0$ models the overdispersion. The model was applied on only one tile with a width of 120 kb, reproducing Figure 2a of Smallwood et al. [95]. Further parameter details are given in Table 4.1.

### 4.8.2 Application to DNA methylation data

Figure 4.13 reproduces an original figure by Smallwood et al. [95], showing the fit in a 120 kb section of chromosome 6. We modeled this 120 kb section with GenoGAM using a quasi-binomial model, 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 (Equation 4.7). Smoothing and dispersion parameters were determined by cross-validation (Section 4.4.4). The GenoGAM fit was consistent with the original publication [95], but did not rely on manually set window sizes and provided confidence bands (Figure 4.13). As expected, wider confidence bands were obtained in regions of sparse data and tighter bands in regions with more data (Figure 4.13).

## 4.9 Discussion and Conclusion

We have introduced a generic framework based on generalized additive models to model ChIP-Seq data, made possible by providing a scalable algorithm that can fit GAMs to large longitudinal data, in our case whole chromosomes at base-pair resolution. GAMs offer a powerful method to smooth, correct, and analyze ChIP-Seq data, because of a very simple way to analyze even complex factorial designs. Unfortunately, the runtime complexity of a single GAM is polynomial in the length of the $x$-axis. Applying a GAM to a whole chromosome is thus unfeasible.

Parallelizing a GAM, on the other hand, is a nontrivial task. Our approach is based on splitting the genomic data up into consecutive but overlapping regions, and computing separate models in each region. Strictly speaking, it is incorrect to split up a chromosome

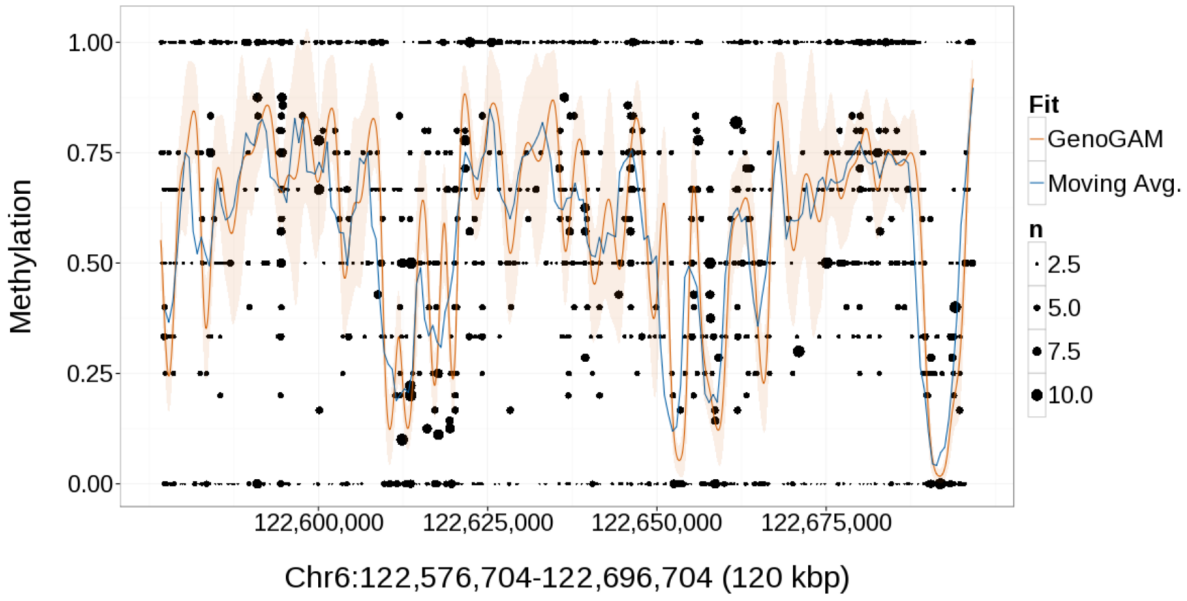



Figure 4.13: **Application to DNA methylation data.** Estimated DNA methylation rates in a 120 kb region of chromosome 6 of the mouse genome (cf. Smallwood et al. [95]). Shown are the data for bulk embryonic mouse stem cells grown in serum; ratios of methylated counts for each CpG position (black dots), with point size proportional to the number of reads. The estimated rates are shown for the moving average approach [95] of 3,000 bp bins in 600 bp steps (blue line) and for GenoGAM (orange line) with 95% confidence band (ribbon).

into multiple regions and fit one model per region, because the data points on the far left and far right end of a chromosome are, mathematically, not independent. They are however what we call "almost independent", in that they do not influence the final estimators notably. Our approach thus delivers an approximation of the "correct" model, with a practically imperceptible amount of error [46].

Smoothing and dispersion parameters were obtained by cross-validation, that is, they were fitted for the accuracy in predicting unseen data. This criterion turned out to provide useful values of smoothing and dispersion for inference, since we obtain signal peaks close to actual binding sites of transcription factors when these are known, at least as close as other dedicated tools. Moreover, this criterion led to reasonable uncertainty estimates because confidence bands of the fits were found to be only slightly conservative.

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 our analyses the samples within an experiment were all similar enough to estimate the parameters globally.

Using genome-wide GAMs offers a number of advantages. First, we flexibly model factorial designs, as well as replicates with different sequencing depths using size factors as offsets. Second, applying GAMs yields confidence bands as a measure of local uncertainty for the estimated rates. We showed how these can be the basis for computing point-wise and region-wise $p$-values. Third, the output of GAMs are analytically differentiable smooth functions, allowing flexible downstream analysis. We showed how peak calling can be elegantly handled by using the first and second derivatives. Fourth, various link functions and distributions can be used, providing the possibility of modelling a wide range of genomic data beyond ChIP-Seq, as we illustrated with a first application on DNA methylation.

On the human genome, there are still two limitations, mentioned in Section 4.6.2: the necessity to analyze the data one chromosome at a time and the rather long runtime of more than 24 hours. Nonetheless, we here established and presented the framework and its statistical properties. Our group currently works on extending GenoGAM to its full potential by overcoming both these limitations.

In conclusion, we foresee GenoGAM as a generic method for the analysis of genome-wide assays.

# Chapter 5

# Summary, Conclusion, and Outlook

Leo Breiman [21] described the two cultures of statistical modeling, users of classical stochastic data models, and users of algorithmic models (machine learning). Data models assume that the data is being generated by a specific stochastic model, often parametrized by a small number of parameters. Linear regression is a simple example of a data model. On the other hand, algorithmic models treat the data generating process as a "black box", and focus solely on a good prediction of the response variable.

While algorithmic models (e.g., bagged classification trees) are often better in terms of predictive quality, most lack the diagnostic outputs that statistical models deliver. Breiman also argues that as data becomes more intricate, the statistical models become more complex as well, thereby losing their advantage of simplicity and interpretability. He further argues for the adaptation of algorithmic models in statistical modeling, primarily because of their superior prediction performance.

However, especially in medical applications, practitioners often need interpretable models. In these cases, algorithmic "black box" models are an undesirable solution. In this thesis, I therefore examined two problems of complex data, and showed that with some improvements, it is still possible to compute statistical data models on them.

In Chapter 3, we explored pedigree data, and developed an EM algorithm that reduces the complexity of the Maximum-Likelihood estimation from exponential to linear regarding the family sizes. We developed a tool that can be applied by physicians to quantify the familial risk for CRC based on a pedigree with disease history.

In Chapter 4, we developed a method that allows computing a GAM on an $x$-axis of arbitrary length, by reducing the complexity from polynomial to linear and implementing a parallelized execution of sequential models. This approach delivers a statistically sound error correction for NGS experiments such as ChIP-Seq, and provides a unified framework

that also covers further downstream analyses.

The specific advantages of data models over algorithmic models in this thesis were (a) the estimation of an interpretable parameter for risk increase in Chapter 3, and (b) allowing the computation of confidence bands and region-wise significance tests in Chapter 4. The key contributions from the statistical/methodical side are therefore in both chapters the development of an efficient estimation algorithm that scales linearly. As for the contributions from the experimental/applied side, in Chapter 3 we built a calculator for estimating familial CRC risk that can directly be applied by physicians. In Chapter 4 we created a software tool for ChIP-Seq analysis that can be applied by bioinformaticians to unify their analysis pipeline into one general framework.

This thesis presented two implementations of algorithms that can efficiently handle large, modern data sets. This demonstrated that revisiting and improving established methods (Section 2.5) is a fruitful approach toward working with data sets that, due to their size and complexity, can not be analyzed with classical methods anymore.

Future research can continue in this direction, and develop new or improve existing algorithms that enable us to extract as much information as possible from large or complex data sets.

In conclusion, I agree with Thomas Heinis [46], who said that the time has come to abandon the need for mathematically exact solutions and settle for approximations with tight enough error bounds, if doing so helps reduce the complexity of a solution, ideally to linear time. To paraphrase John Tukey, “an approximate answer to the right question is better than an exact answer to the wrong question” [100].

# Appendix A

# Supplementary Material for Chapter 3

## A.1 Efficient computation of likelihoods and marginalizations

Since we assume independence among families, the complete likelihood $L(\theta; x, z)$ can be factorized into a product of $N$ *family likelihoods* [97, Eq. 1.4]. If one denotes the families in a data set by $I = 1, \ldots, N$ and their members by $d = 1, \ldots, D_I$, the index $i$ becomes a combined index $I, d$ from the family index and the member index. We can further define the sub-vectors $x_I$ and $z_I$ to be the observed and latent data for only family $I$. For example, using this notation, $z_1$ is now a vector of risk statuses for family 1, and not the risk status of just the first observation. The risk status for the second member in family 1 would be $z_{1,2}$.

Equation 3.2 then becomes

$$
\begin{aligned}
L(\theta; x, z) &= \prod_{I=1}^{N} f(x_I, z_I) = \prod_{I=1}^{N} \mathbb{P}(z_I) \cdot f(x_I | z_I) \\
&= \prod_{I=1}^{N} \Big\{ \prod_{d \in F_I} \mathbb{P}(z_{I,d}) \cdot \prod_{d \notin F_I} \mathbb{P}(z_{I,d} | z_{\male_{I,d}}, z_{\female_{I,d}}) \cdot \prod_{d=1}^{D_I} f(x_{I,d} | z_{I,d}) \Big\} \\
&= \prod_{I=1}^{N} \Big\{ \prod_{d \in F_I} p_1^{z_{I,d}} (1-p_1)^{1-z_{I,d}} \cdot \prod_{d \notin F_I} \tilde{p}_{I,d}^{z_{I,d}} (1-\tilde{p}_{I,d})^{1-z_{I,d}} \\
&\quad \cdot \prod_{d=1}^{D_I} [k \lambda^k t_{I,d}^{k-1} \alpha^{z_{I,d}} \beta^{m_{I,d}}]^{c_{I,d}} \exp(-(t_{I,d}\lambda)^k \alpha^{z_{I,d}} \beta^{m_{I,d}}) \Big\}
\end{aligned}
$$

This notation now allows for computationally elegant marginalizations:

## Summation in the marginalized likelihood in Equation 3.6

Keeping the notation for families and family members, we can rewrite Equation 3.6 into a more efficient marginalization:

$$
\begin{aligned}
L(\theta; x) &= \sum_{z} L(\theta; x, z) \\
&= \sum_{z} \prod_{I=1}^{N} L(\theta; x_I, z_I) \\
&= \sum_{z_1} \dots \sum_{z_N} L(\theta; x_1, z_1) \cdot \ldots \cdot L(\theta; x_N, z_N) \\
&= \sum_{z_1} L(\theta; x_1, z_1) \cdot \ldots \cdot \sum_{z_N} L(\theta; x_N, z_N) \\
&= \prod_{I=1}^{N} \sum_{z_I} L(\theta; x_I, z_I) \\
&= \prod_{I=1}^{N} \sum_{z_I} f(x_I | z_I, \theta)\ \mathbb{P}(z_I)
\end{aligned}
$$

This way, we do not sum over $2^n$ possible values for $z$, but instead $\prod_{I=1}^{N} 2^{D_I}$ values, where $D_I$ is the number of family members in family $I$.

### Marginalizing the risk carrier probability $T_i^{(t)}$ from Equation 3.7

The marginalization when computing $T_i^{(t)} \equiv T_{I,d}^{(t)}$ can happen more efficiently, summing only over one specific family. Since a $Z_{I,d}$ is independent of all $x$ outside of its respective family's $x_I$, we can replace the condition on $x$ by $x_I$:

$$\begin{aligned} T_{I,d}^{(t)} &= \mathbb{E}_{Z|x,\theta^{(t)}}(Z_{I,d}) \\ &= \mathbb{E}_{Z_{I,d}|x,\theta^{(t)}}(Z_{I,d}) && \text{(Equation 3.8)} \\ &= \mathbb{E}_{Z_{I,d}|x_I,\theta^{(t)}}(Z_{I,d}) \\ &= \mathbb{P}(Z_{I,d} = 1|x_I, \theta^{(t)}) \\ &= \sum_{z_I} z_{I,d} \mathbb{P}(z_I|x_I, \theta^{(t)}) \end{aligned}$$

Therefore, the marginalizations of the factor graph can be efficiently computed for each family separately and combined at the end.

## A.2 All messages of the sum-product algorithm in Figure 3.3

Here we show a detailed derivation of all messages for the sum-product algorithm in Figure 3.3. The messages are computed iteratively in 8 steps. Figure A.1 illustrates the messages computed in each step.

**Step 1** In step 1, only the "outermost" messages can be computed, since only they are not depending on any other messages. In particular, the following messages are computed:

- $\mu_{\phi_1 \to z_1}(z_1) = \phi_1(z_1)$
- $\mu_{\phi_2 \to z_2}(z_2) = \phi_2(z_2)$
- $\mu_{\phi_3 \to z_3}(z_3) = \phi_3(z_3)$
- $\mu_{\phi_4 \to z_4}(z_4) = \phi_4(z_4)$
- $\mu_{z_7 \to \phi_{789}}(z_7) = 1$
- $\mu_{z_8 \to \phi_{789}}(z_8) = 1$
- $\mu_{z_9 \to \phi_{789}}(z_9) = 1$

The definitions of each factor, e.g. $\phi_1(z_1)$ are available in Supplementary Material 3.3.8

**Step 2** In step 2 (see Figure A.1b), we have all information necessary to compute the following four messages:

- $\mu_{z_1 \to \phi_5}(z_1) = \mu_{\phi_1 \to z_1}(z_1)$
- $\mu_{z_2 \to \phi_5}(z_2) = \mu_{\phi_2 \to z_2}(z_2)$
- $\mu_{z_3 \to \phi_6}(z_3) = \mu_{\phi_3 \to z_3}(z_3)$
- $\mu_{z_4 \to \phi_6}(z_4) = \mu_{\phi_4 \to z_4}(z_4)$

**Step 3** Now we can compute the messages *to* the parents $z_5$ and $z_6$:

- $\mu_{\phi_5 \to z_5}(z_5) = \sum_{z_1} \sum_{z_2} (\phi_5(z_1, z_2, z_5) \cdot \mu_{z_1 \to \phi_5}(z_1) \mu_{z_2 \to \phi_5}(z_2))$
- $\mu_{\phi_6 \to z_6}(z_6) = \sum_{z_3} \sum_{z_4} (\phi_6(z_3, z_4, z_6) \cdot \mu_{z_3 \to \phi_6}(z_3) \mu_{z_4 \to \phi_6}(z_4))$

**Step 4** In step 4, we can only obtain two new messages:

- $\mu_{z_5 \to \phi_{789}}(z_5) = \mu_{\phi_5 \to z_5}(z_5)$
- $\mu_{z_6 \to \phi_{789}}(z_6) = \mu_{\phi_6 \to z_6}(z_6)$

**Step 5** In step 5, we now have all incoming messages to $\phi_{789}$ available, which means we can compute all outgoing messages from $\phi_{789}$:

- $\mu_{\phi_{789} \to z_5}(z_5) = \sum_{z_6} \sum_{z_7} \sum_{z_8} \sum_{z_9} \Big( \phi_{789}(z_5, z_6, z_7, z_8, z_9) \cdot \mu_{z_6 \to \phi_{789}}(z_6) \mu_{z_7 \to \phi_{789}}(z_7) \mu_{z_8 \to \phi_{789}}(z_8) \mu_{z_9 \to \phi_{789}}(z_9) \Big)$
- $\mu_{\phi_{789} \to z_6}(z_6) = \sum_{z_5} \sum_{z_7} \sum_{z_8} \sum_{z_9} \Big( \phi_{789}(z_5, z_6, z_7, z_8, z_9) \cdot \mu_{z_5 \to \phi_{789}}(z_5) \mu_{z_7 \to \phi_{789}}(z_7) \mu_{z_8 \to \phi_{789}}(z_8) \mu_{z_9 \to \phi_{789}}(z_9) \Big)$
- $\mu_{\phi_{789} \to z_7}(z_7) = \sum_{z_5} \sum_{z_6} \sum_{z_8} \sum_{z_9} \Big( \phi_{789}(z_5, z_6, z_7, z_8, z_9) \cdot \mu_{z_5 \to \phi_{789}}(z_5) \mu_{z_6 \to \phi_{789}}(z_6) \mu_{z_8 \to \phi_{789}}(z_8) \mu_{z_9 \to \phi_{789}}(z_9) \Big)$
- $\mu_{\phi_{789} \to z_8}(z_8) = \sum_{z_5} \sum_{z_6} \sum_{z_7} \sum_{z_9} \Big( \phi_{789}(z_5, z_6, z_7, z_8, z_9) \cdot \mu_{z_5 \to \phi_{789}}(z_5) \mu_{z_6 \to \phi_{789}}(z_6) \mu_{z_7 \to \phi_{789}}(z_7) \mu_{z_9 \to \phi_{789}}(z_9) \Big)$
- $\mu_{\phi_{789} \to z_9}(z_9) = \sum_{z_5} \sum_{z_6} \sum_{z_7} \sum_{z_8} \Big( \phi_{789}(z_5, z_6, z_7, z_8, z_9) \cdot \mu_{z_5 \to \phi_{789}}(z_5) \mu_{z_6 \to \phi_{789}}(z_6) \mu_{z_7 \to \phi_{789}}(z_7) \mu_{z_8 \to \phi_{789}}(z_8) \Big)$

**Step 6** We can now start computing the *upward* messages from $\phi_{789}$:

- $\mu_{z_5 \to \phi_5}(z_5) = \mu_{\phi_{789} \to z_5}(z_5)$

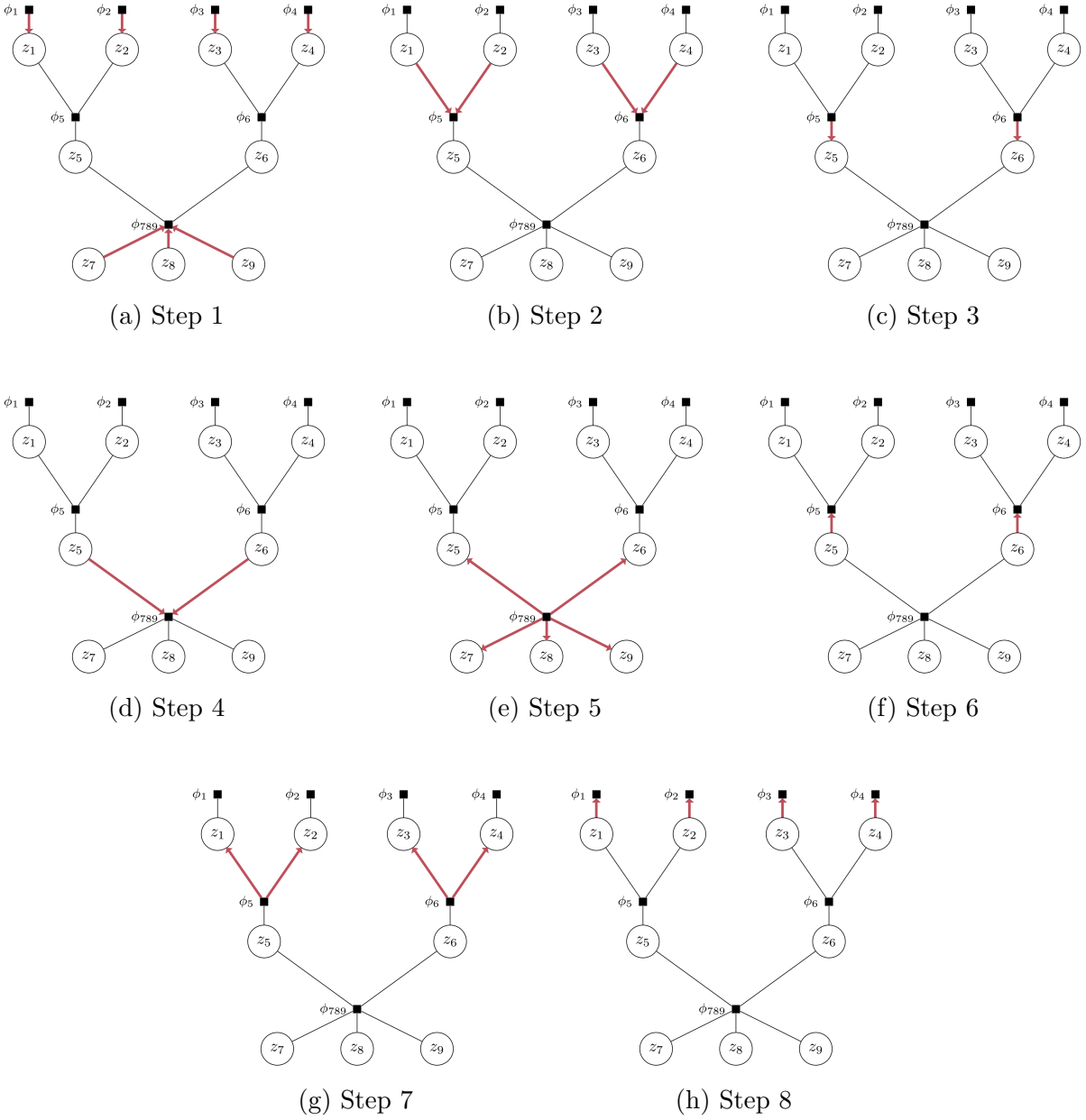
- $\mu_{z_6 \to \phi_6}(z_6) = \mu_{\phi_{789} \to z_6}(z_6)$

**Step 7** Here, we compute the upward messages from the factors $\phi_5$ and $\phi_6$:

- $\mu_{\phi_5 \to z_1}(z_1) = \sum_{z_2} \sum_{z_5} (\phi_5(z_1, z_2, z_5) \cdot \mu_{z_2 \to \phi_5}(z_2) \mu_{z_5 \to \phi_5}(z_5))$
- $\mu_{\phi_5 \to z_2}(z_2) = \sum_{z_1} \sum_{z_5} (\phi_5(z_1, z_2, z_5) \cdot \mu_{z_1 \to \phi_5}(z_1) \mu_{z_5 \to \phi_5}(z_5))$
- $\mu_{\phi_6 \to z_3}(z_3) = \sum_{z_4} \sum_{z_6} (\phi_6(z_3, z_4, z_6) \cdot \mu_{z_4 \to \phi_6}(z_4) \mu_{z_6 \to \phi_6}(z_6))$
- $\mu_{\phi_6 \to z_4}(z_4) = \sum_{z_3} \sum_{z_6} (\phi_6(z_3, z_4, z_6) \cdot \mu_{z_3 \to \phi_6}(z_3) \mu_{z_6 \to \phi_6}(z_6))$

**Step 8** In the last step, we can compute the messages from the founder variables to their respective factors. For the marginalization step in the end, we do not need these messages, strictly speaking. For the sake of completeness, however, we show their computation nonetheless:

- $\mu_{z_1 \to \phi_1}(z_1) = \mu_{\phi_5 \to z_1}(z_1)$
- $\mu_{z_2 \to \phi_2}(z_2) = \mu_{\phi_5 \to z_2}(z_2)$
- $\mu_{z_3 \to \phi_3}(z_3) = \mu_{\phi_6 \to z_3}(z_3)$
- $\mu_{z_4 \to \phi_4}(z_4) = \mu_{\phi_6 \to z_4}(z_4)$




(a) Step 1 (b) Step 2 (c) Step 3

(d) Step 4 (e) Step 5 (f) Step 6

(g) Step 7 (h) Step 8

Figure A.1: Message passing during the sum-product algorithm. The messages are computed iteratively. Once all but one incoming message are calculated for a given node, a message is calculated and sent across the remaining edge. Note that by Kschischang et al. [62], the actual order in which the messages are calculated is irrelevant for the result.

# Appendix B

# Supplementary Material for Chapter 4

## B.1 ChIP-Seq library preparation, sequencing and read alignment of the yeast TFIIB data set

ChIP-Seq for TFIIB was performed essentially as described previously [92] 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 $\mu$l of IgG SepharoseTM 6 Fast Flow beads (GE Healthcare) for 3 hours at 4°C on a turning wheel. 30 $\mu$l of Input sample was taken before immunoprecipitation and stored at 4°C. IP and Input samples underwent reverse cross-linking for 2 hours with Proteinase K at 65°C and purified using Quiagen MinElute Kit. Samples were digested with 2.5 $\mu$l 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 $\mu$l H2O. ChIP-Seq libraries were prepared using a NEB Next library preparation kit following manufacturer's instructions using the complete 50 $\mu$l as input. 2 $\mu$l of 1.7 $\mu$M adapters containing a GGAT barcode and 2 $\mu$l of a 0.25 $\mu$M 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. Next, 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 [40]. The reads were then demultiplexed, quality-trimmed (Fastq Quality Filter), and mapped with Bowtie 0.12.7 [66] to the SacCer2 genome assembly (Bowtie options: `-q -p 4 -S --sam-nohead -phred33-quals`).

## B.2 Data processing of the H3K4me3 data set

Raw sequencing files (H3K4ME3_Full_length_Set1_Rep_1.fastq, H3K4ME3_Full_length_Set1_Rep_2.fastq, H3K4ME3_aa762-1080_Set1_Rep_1.fastq, and 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 [29] (version 2.4.0, default parameters). Reads with ambiguous mapping were removed using samtools [67] (version 1.2, option `-q 255`). Gene boundaries were obtained from the *S. cerevisiae* genome annotation R64.1.1, restricting gff file entries to type "gene".

# Bibliography


[1] R. Abbi, E. El-Darzi, C. Vasilakis, and P. Millard. Analysis of stopping criteria for the EM algorithm in the context of patient grouping according to length of stay. In *2008 4th International IEEE Conference Intelligent Systems*, volume 1, pages 3–9. IEEE, 2008.

[2] D. Aird, M. G. Ross, W.-S. Chen, M. Danielsson, T. Fennell, C. Russ, D. B. Jaffe, C. Nusbaum, and A. Gnirke. Analyzing and minimizing pcr amplification bias in illumina sequencing libraries. *Genome biology*, 12(2):1, 2011.

[3] I. Albert, T. N. Mavrich, L. P. Tomsho, J. Qi, S. J. Zanton, S. C. Schuster, and B. F. Pugh. Translational and rotational settings of H2A.Z nucleosomes across the Saccharomyces cerevisiae genome. *Nature*, 446(7135):572–6, mar 2007.

[4] C. F. Aliferis, I. Tsamardinos, A. R. Statnikov, and L. E. Brown. Causal explorer: A causal probabilistic network learning toolkit for biomedical discovery. In *METMBS*, volume 3, pages 371–376, 2003.

[5] B. Alipanahi, A. Delong, M. T. Weirauch, and B. J. Frey. Predicting the sequence specificities of DNA-and RNA-binding proteins by deep learning. *Nature biotechnology*, 2015.

[6] Amazon. EC2 instance pricing. `https://aws.amazon.com/ec2/pricing/on-demand/`. Accessed: 2016-12-02.

[7] G. M. Amdahl. Validity of the single processor approach to achieving large scale computing capabilities. In *Proceedings of the April 18-20, 1967, spring joint computer conference*, pages 483–485. ACM, 1967.

[8] S. Anders and W. Huber. Differential expression analysis for sequence count data. *Genome biology*, 11(10):1, 2010.

[9] M. R. Barnes and I. C. Gray. *Bioinformatics for geneticists*. John Wiley & Sons, 2003.

[10] A. Barski, S. Cuddapah, K. Cui, T.-Y. Roh, D. E. Schones, Z. Wang, G. Wei, I. Chepelev, and K. Zhao. High-resolution profiling of histone methylations in the human genome. *Cell*, 129(4):823–37, 2007.

[11] A. D. Basehoar, S. J. Zanton, and B. F. Pugh. Identification and distinct regulation of yeast TATA box-containing genes. *Cell*, 116(5):699–709, 2004.

[12] L. E. Baum. An equality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes. *Inequalities*, 3:1–8, 1972.

[13] S. Behjati and P. S. Tarpey. What is next generation sequencing? *Archives of disease in childhood-Education & practice edition*, 98(6):236–238, 2013.

[14] C. J. Bélisle. Convergence theorems for a class of simulated annealing algorithms on $\mathbb{R}^d$. *Journal of Applied Probability*, 29(04):885–895, 1992.

[15] N. M. Belonogova and T. I. Axenovich. Optimal peeling order for pedigrees with incomplete genotypic information. *Computational biology and chemistry*, 31(3):173–177, 2007.

[16] Y. Benjamini and Y. Hochberg. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 289–300, 1995.

[17] Y. Benjamini and D. Yekutieli. The control of the false discovery rate in multiple testing under dependency. *Annals of statistics*, pages 1165–1188, 2001.

[18] J. L. Bermejo and K. Hemminki. Familial risk of cancer shortly after diagnosis of the first familial tumor. *Journal of the National Cancer Institute*, 97(21):1575–1579, 2005.

[19] R. Bourgon, R. Gentleman, and W. Huber. Independent filtering increases detection power for high-throughput experiments. *Proceedings of the National Academy of Sciences*, 107(21):9546–9551, 2010.

[20] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[21] L. Breiman et al. Statistical modeling: The two cultures (with comments and a rejoinder by the author). *Statistical science*, 16(3):199–231, 2001.

[22] H. Brenner, M. Hoffmeister, and U. Haug. Family history and age at initiation of colorectal cancer screening. *The American journal of gastroenterology*, 103(9):2326–2331, 2008.

[23] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16(5):1190–1208, 1995.

[24] C. Cannings, E. Thompson, and M. Skolnick. Probability functions on complex pedigrees. *Advances in Applied Probability*, pages 26–61, 1978.

[25] P. Collas. The current state of chromatin immunoprecipitation. *Molecular biotechnology*, 45(1):87–100, 2010.

[26] C. De Boor. *A practical guide to splines*, volume 27. Springer-Verlag New York, 1978.

[27] F. Dellaert. The expectation maximization algorithm. *Technical Report*, 2002.

[28] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38, 1977.

[29] A. Dobin, C. A. Davis, F. Schlesinger, J. Drenkow, C. Zaleski, S. Jha, P. Batut, M. Chaisson, and T. R. Gingeras. STAR: ultrafast universal RNA-seq aligner. *Bioinformatics (Oxford, England)*, 29(1):15–21, 2013.

[30] P. H. Eilers and B. D. Marx. Flexible smoothing with B-splines and penalties. *Statistical science*, pages 89–102, 1996.

[31] R. C. Elston and J. Stewart. A general model for the genetic analysis of pedigree data. *Human heredity*, 21(6):523–542, 1971.

[32] ENCODE Project Consortium and others. An integrated encyclopedia of DNA elements in the human genome. *Nature*, 489(7414):57–74, 2012.

[33] H. Esmaeilzadeh, E. Blem, R. S. Amant, K. Sankaralingam, and D. Burger. Power challenges may end the multicore era. *Communications of the ACM*, 56(2):93–102, 2013.

[34] H. Failmezger, E. Dursun, T. Schroeder, A. Krug, and A. Tresch. Quantification of deterministic and stochastic cell fate components using hidden factor graph models. Submitted to PLoS Comp Biol.

[35] J. Friedman, T. Hastie, and R. Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics Springer, Berlin, 2001.

[36] A. E. Gelfand and A. F. Smith. Sampling-based approaches to calculating marginal densities. *Journal of the American statistical association*, 85(410):398–409, 1990.

[37] R. C. Gentleman, V. J. Carey, D. M. Bates, B. Bolstad, M. Dettling, S. Dudoit, B. Ellis, L. Gautier, Y. Ge, J. Gentry, K. Hornik, T. Hothorn, W. Huber, S. Iacus, R. Irizarry, F. Leisch, C. Li, M. Maechler, A. J. Rossini, G. Sawitzki, C. Smith, G. Smyth, L. Tierney, J. Y. H. Yang, and J. Zhang. Bioconductor: open software development for computational biology and bioinformatics. *Genome biology*, 5(10):R80, 2004.

[38] C. J. Geyer and E. A. Thompson. Constrained Monte Carlo maximum likelihood for dependent data. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 657–699, 1992.

[39] Z. Ghahramani. An introduction to hidden Markov models and Bayesian networks. *International Journal of Pattern Recognition and Artificial Intelligence*, 15(01):9–42, 2001.

[40] J. Goecks, A. Nekrutenko, and J. Taylor. Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. *Genome Biology*, 11(8):R86, jan 2010.

[41] S. Goodwin, J. D. McPherson, and W. R. McCombie. Coming of age: ten years of next-generation sequencing technologies. *Nature Reviews Genetics*, 17(6):333–351, 2016.

[42] C. E. Grant, T. L. Bailey, and W. S. Noble. FIMO: scanning for occurrences of a given motif. *Bioinformatics (Oxford, England)*, 27(7):1017–8, apr 2011.

[43] S. W. Guo and E. Thompson. A Monte Carlo method for combined segregation and linkage analysis. *American journal of human genetics*, 51(5):1111, 1992.

[44] E. Half, D. Bercovich, and P. Rozen. Familial adenomatous polyposis. *Orphanet journal of rare diseases*, 4(1):22, 2009.

[45] T. Hastie, R. Tibshirani, et al. Generalized additive models. *Statistical science*, 1(3):297–310, 1986.

[46] T. Heinis. Data analysis: approximation aids handling of big data. *Nature*, 515(7526):198, 2014.

[47] C. A. Hoare. Quicksort. *The Computer Journal*, 5(1):10–16, 1962.

[48] Y. Hochberg. A sharper Bonferroni procedure for multiple tests of significance. *Biometrika*, 75(4):800–802, 1988.

[49] D. Houle, D. R. Govindaraju, and S. Omholt. Phenomics: the next challenge. *Nature Reviews Genetics*, 11(12):855–866, 2010.

[50] IBM. What is big data? `https://www-01.ibm.com/software/data/bigdata/what-is-big-data.html`, 2016. [Online; accessed 11-December-2016].

[51] M. M. Ibrahim, S. A. Lacadie, and U. Ohler. JAMM: a peak finder for joint analysis of NGS replicates. *Bioinformatics (Oxford, England)*, 31(1):48–55, 2015.

[52] A. T. Ihler, W. F. John III, and A. S. Willsky. Loopy belief propagation: Convergence and effects of message errors. *Journal of Machine Learning Research*, 6(May):905–936, 2005.

[53] International Human Genome Sequencing Consortium and others. Finishing the euchromatic sequence of the human genome. *Nature*, 431(7011):931–945, 2004.

[54] G. P. Jarvik. Complex segregation analyses: uses and limitations. *The American Journal of Human Genetics*, 63(4):942–946, 1998.

[55] D. S. Johnson, A. Mortazavi, R. M. Myers, and B. Wold. Genome-wide mapping of in vivo protein-DNA interactions. *Science (New York, N.Y.)*, 316(5830):1497–502, 2007.

[56] G. Juve, E. Deelman, K. Vahi, G. Mehta, B. Berriman, B. P. Berman, and P. Maechling. Scientific workflow applications on Amazon EC2. In *2009 5th IEEE International Conference on E-Science Workshops*, pages 59–66. IEEE, 2009.

[57] P. Kaatsch, C. Spix, S. Hentschel, A. Katalinic, S. Luttmann, C. Stegmaier, S. Caspritz, J. Cernaj, A. Ernst, J. Folkerts, et al. Krebs in Deutschland 2009/2010. *Robert Koch-Institut*, 2013.

[58] D. E. Knuth. *The art of computer programming: sorting and searching*, volume 3. Pearson Education, 1998.

[59] F. T. Kolligs, A. Crispin, A. Munte, A. Wagner, U. Mansmann, and B. Göke. Risk of advanced colorectal neoplasia according to age and gender. *PloS one*, 6(5):e20076, 2011.

[60] D. Kostrewa, M. E. Zeller, K.-J. Armache, M. Seizl, K. Leike, M. Thomm, and P. Cramer. Rna polymerase ii–tfiib structure and mechanism of transcription initiation. *Nature*, 462(7271):323–330, 2009.

[61] L. Kruglyak, M. J. Daly, M. P. Reeve-Daly, and E. S. Lander. Parametric and nonparametric linkage analysis: a unified multipoint approach. *American journal of human genetics*, 58(6):1347, 1996.

[62] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on information theory*, 47(2):498–519, 2001.

[63] S. G. Landt, G. K. Marinov, A. Kundaje, P. Kheradpour, F. Pauli, S. Batzoglou, B. E. Bernstein, P. Bickel, J. B. Brown, P. Cayting, et al. ChIP-seq guidelines and practices of the ENCODE and modENCODE consortia. *Genome research*, 22(9):1813–1831, 2012.

[64] K. Lange. *Mathematical and Statistical Methods for Genetic Analysis*. Springer, 1997.

[65] K. Lange and R. Elston. Extensions to pedigree analysis I. Likehood calculations for simple and complex pedigrees. *Human Heredity*, 25(2):95–105, 1974.

[66] B. Langmead, C. Trapnell, M. Pop, and S. L. Salzberg. Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome Biology*, 10(3):R25, jan 2009.

[67] H. Li, B. Handsaker, A. Wysoker, T. Fennell, J. Ruan, N. Homer, G. Marth, G. Abecasis, and R. Durbin. The Sequence Alignment/Map format and SAMtools. *Bioinformatics (Oxford, England)*, 25(16):2078–9, aug 2009.

[68] L. Liu, Y. Li, S. Li, N. Hu, Y. He, R. Pong, D. Lin, L. Lu, and M. Law. Comparison of next-generation sequencing systems. *BioMed Research International*, 2012, 2012.

[69] M. I. Love, W. Huber, and S. Anders. Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2. *Genome Biology*, 15(12):550, 2014.

[70] A. T. Lun and G. K. Smyth. De novo detection of differentially bound regions for ChIP-seq data using peaks and windows: controlling error rates correctly. *Nucleic acids research*, 42(11):e95–e95, 2014.

[71] U. Mansmann, J. Stausberg, J. Engel, P. Heussner, B. Birkner, and C. Maar. Familien schützen und stärken – Umgang mit familiärem Darmkrebs. eine Pilotstudie zur Inzidenz von Risikoclustern und zur Möglichkeit ihrer Detektion. *Der Gastroenterologe*, 7:271–272, 2012.

[72] K. Markianos, M. J. Daly, and L. Kruglyak. Efficient multipoint linkage analysis through reduction of inheritance space. *The American Journal of Human Genetics*, 68(4):963–977, 2001.

[73] G. Marra and S. N. Wood. Coverage properties of confidence intervals for generalized additive model components. *Scandinavian Journal of Statistics*, 39(1):53–74, 2012.

[74] A. Mathelier, X. Zhao, A. W. Zhang, F. Parcy, R. Worsley-Hunt, D. J. Arenillas, S. Buchman, C.-y. Chen, A. Chou, H. Ienasescu, J. Lim, C. Shyr, G. Tan, M. Zhou, B. Lenhard, A. Sandelin, and W. W. Wasserman. JASPAR 2014: an extensively expanded and updated open-access database of transcription factor binding profiles. *Nucleic acids research*, 42(Database issue):D142–7, 2014.

[75] G. McLachlan and T. Krishnan. *The EM algorithm and extensions*. John Wiley & Sons, 2007.

[76] R. B. Mendelsohn and A. J. Markowitz. Hereditary colon cancer. *European Gastroenterology and Hepatology Review*, 7:251–256, 2011.

[77] M. L. Metzker. Sequencing technologies—the next generation. *Nature reviews genetics*, 11(1):31–46, 2010.

[78] M. Mezard and A. Montanari. *Information, physics, and computation*. Oxford University Press, 2009.

[79] D. Michie. Memo functions and machine learning. *Nature*, 218(5136):19–22, 1968.

[80] G. E. Moore. Cramming more components onto integrated circuits, reprinted from electronics, volume 38, number 8, april 19, 1965, pp. 114 ff. *IEEE Solid-State Circuits Newsletter*, 3(20):33–35, 2006.

[81] D. Nasseh, J. Engel, U. Mansmann, W. Tretter, and J. Stausberg. Matching study to registry data: maintaining data privacy in a study on family based colorectal cancer. *Studies in health technology and informatics*, 205:808–812, 2014.

[82] J. A. Nelder and R. Mead. A simplex method for function minimization. *The computer journal*, 7(4):308–313, 1965.

[83] H. H. Ng, F. Robert, R. A. Young, and K. Struhl. Targeted recruitment of Set1 histone methylase by elongating Pol II provides a localized mark and memory of recent transcriptional activity. *Molecular cell*, 11(3):709–719, 2003.

[84] P. C. Ng and E. F. Kirkness. Whole genome sequencing. In *Genetic variation*, pages 215–226. Springer, 2010.

[85] T. D. Nielsen and F. V. Jensen. *Bayesian networks and decision graphs*. Springer Science & Business Media, 2009.

[86] W. P. Petersen and P. Arbenz. *Introduction to parallel computing*. Oxford University Press, 2004.

[87] E. Pettersson, J. Lundeberg, and A. Ahmadian. Generations of sequencing technologies. *Genomics*, 93(2):105–111, 2009.

[88] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2016.

[89] N. U. Rashid, P. G. Giresi, J. G. Ibrahim, W. Sun, and J. D. Lieb. ZINBA integrates local covariates with DNA-seq data to identify broad and narrow regions of enrichment, even within amplified genomic regions. *Genome Biology*, 12(7):R67, 2011.

[90] A. Rieger and U. R. Mansmann. Bayesian prediction of being a colorectal cancer risk family. Manuscript in preparation.

[91] G. Robertson, M. Hirst, M. Bainbridge, M. Bilenky, Y. Zhao, T. Zeng, G. Euskirchen, B. Bernier, R. Varhol, A. Delaney, et al. Genome-wide profiles of STAT1 DNA association using chromatin immunoprecipitation and massively parallel sequencing. *Nature methods*, 4(8):651–657, 2007.

[92] D. Schulz, B. Schwalb, A. Kiesel, C. Baejen, P. Torkler, J. Gagneur, J. Soeding, and P. Cramer. Transcriptome Surveillance by Selective Termination of Noncoding RNA Synthesis. *Cell*, 155(5):1075–1087, 2013.

[93] G. Schweikert, B. Cseke, T. Clouaire, A. Bird, and G. Sanguinetti. MMDiff: quantitative testing for shape changes in ChIP-Seq data sets. *BMC genomics*, 14(1):826, 2013.

[94] M. Sipser. *Introduction to the Theory of Computation*, volume 2. Thomson Course Technology Boston, 2006.

[95] S. A. Smallwood, H. J. Lee, C. Angermueller, F. Krueger, H. Saadeh, J. Peat, S. R. Andrews, O. Stegle, W. Reik, and G. Kelsey. Single-cell genome-wide bisulfite sequencing for assessing epigenetic heterogeneity. *Nature methods*, 11(8):817–820, 2014.

[96] The Munich Cancer Registry. `http://www.tumorregister-muenchen.de/en/index.php`, 2016. [Online; accessed 11-October-2016].

[97] E. A. Thompson. Statistical inference from genetic data on pedigrees. In *NSF-CBMS regional conference series in probability and statistics*. JSTOR, 2000.

[98] E. A. Thompson and R. Shaw. Pedigree analysis for quantitative traits: variance components without matrix inversion. *Biometrics*, pages 399–413, 1990.

[99] J. Thornton, G. Westfield, Y. Takahashi, M. Cook, X. Gao, W. A.R., L. J., A. Morgan, J. Jackson, E. Smith, J. Couture, G. Skiniotis, and A. Shilatifard. Context dependency of Set1/ COMPASS-mediated histone H3 Lys4 trimethylation. *Genes & Development*, 28(2):115–120, 2014.

[100] J. W. Tukey. The future of data analysis. *The Annals of Mathematical Statistics*, 33(1):1–67, 1962.

[101] J. Waldo, H. Lin, and L. I. Millett. *Engaging privacy and information technology in a digital age*. National Academies Press Washington, DC, USA, 2007.

[102] R. W. Wedderburn. Quasi-likelihood functions, generalized linear models, and the Gauss—Newton method. *Biometrika*, 61(3):439–447, 1974.

[103] Z. Wei, W. Sun, K. Wang, and H. Hakonarson. Multiple testing in genome-wide association studies via hidden Markov models. *Bioinformatics (Oxford, England)*, 25(21):2802–2808, 2009.

[104] H. Wickham. The split-apply-combine strategy for data analysis. *Journal of Statistical Software*, 40:1–29, 2011.

[105] H. Wickham. *Advanced R*. CRC Press, 2014.

[106] S. Wood. *Generalized additive models: an introduction with R*. CRC press, 2006.

[107] Z. Xu, W. Wei, J. Gagneur, F. Perocchi, S. Clauder-Münster, J. Camblong, E. Guffanti, F. C. O. Stutz, W. Huber, and L. Steinmetz. Bidirectional promoters generate pervasive transcription in yeast. *Nature*, 2009.

[108] H. Yan, J. Evans, M. Kalmbach, R. Moore, S. Middha, S. Luban, L. Wang, A. Bhagwate, Y. Li, Z. Sun, et al. HiChIP: a high-throughput pipeline for integrative analysis of ChIP-Seq data. *BMC bioinformatics*, 15(1):280, 2014.

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

[110] Y. Zhang, T. Liu, C. A. Meyer, J. Eeckhoute, D. S. Johnson, B. E. Bernstein, C. Nusbaum, R. M. Myers, M. Brown, W. Li, et al. Model-based analysis of ChIP-Seq (MACS). *Genome Biology*, 9(9):R137, 2008.

# Acknowledgments

I am very thankful to Prof. Dr. Ulrich Mansmann, Prof. Dr. Achim Tresch and Prof. Dr. Julien Gagneur for giving me the opportunity to work on these projects as well as for their support and supervision during this period.

I am also grateful to the remaining members of my thesis committee, Prof. Dr. Heike Bickeböller, Prof. Dr. Helmut Küchenhoff and Prof. Dr. Volker Schmid, for their time and support.

Many thanks go to Georg Stricker for a very pleasant cooperation on the GenoGAM project, to Anna Rieger for our fruitful discussions on the study design and estimation approaches for the EM algorithm, and to Thomas Uplasnik for his help with the figure design.

I would also like to acknowledge the Graduate School of Quantitative Biosciences Munich (QBM) and the IMPRS at the Max Planck Institute of Plant Breeding Research for financial support during my work on these projects.

And last but not least, I send my sincere gratitute to my parents for their unwavering support and for enabling me to study the most interesting science of statistics.

# Eidesstattliche Versicherung

(Siehe Promotionsordnung vom 12. Juli 2011, § 8 Abs. 2 Pkt. 5)

Hiermit erkläre ich an Eides statt, dass die Dissertation von mir selbstständig, ohne unerlaubte Beihilfe angefertigt ist.

München, 18. April 2017 Alexander Engelhardt