Bioinformatics from Genetic Variants to Methylation

Dissertation

zur Erlangung des Grades eines

Doktor der Naturwissenschaften

der Technischen Universität Dortmund
an der Fakultät für Informatik

von

Christopher Schröder

Dortmund

2018

Tag der mündlichen Prüfung: 9.10.2018

Dekan: Prof. Dr.-Ing. Gernot A. Fink

1. Gutachter: Prof. Dr. Sven Rahmann

2. Gutachter: Prof. Dr. Gunnar W. Klau

Abstract

An important research topic in bioinformatics is the analysis of DNA, the molecule that encodes the genetic information of all organisms. The basis for this is sequencing, a procedure in which the sequence of DNA bases is determined. In addition to the identification of variations in the base sequence itself, advances in sequencing methods and a steady reduction in sequencing costs open up new fields of research: the analysis of functionally relevant non-base-related changes, so-called epigenetics. An important example of such a mechanism is DNA methylation, a process in which methyl groups are added to DNA without altering the sequence itself. Methylation takes place only at specific sites, and the methylation information of human DNA consists of approximately 30 million methylation levels between 0 and 1 in total. This thesis deals with problems and solutions for each phase of DNA methylation analysis.

The most advanced method for detecting DNA methylation based on resolution is Whole-Genome Bisulfite Sequencing (WGBS), a technique that modifies DNA at unmethylated sites. We describe the special in-silico treatment required to process this altered DNA and existing concepts as well as newly developed bioinformatic methods for efficient determination of DNA methylation levels and their further processing with our developed tool `camel`. A common downstream analysis step is the detection of differentially methylated regions (DMRs), for which we have implemented a modification of the widely used method `BSmooth` in order to deal with today's common data sizes.

Setting up and creating new sequencing protocols, e.g., the mentioned WGBS, is complicated and requires adjustments to several parameters. We have developed a method based on a linear program (LP) that can predict the duplicate rate of supersamples. This critical quality measure represents the proportion of redundant data that in most cases needs to be removed from any further analysis. By using our method, it becomes possible to test, adjust and improve parameters for small test libraries only and to estimate the duplication rate for potential full-size samples.

Once the sequencing protocol has been established, the methylation recognition of `camel` can be used as part of automated workflows, such as our `mosquito` workflow. This pipeline processes the generated WGBS samples from the raw data to the degree of methylation, including all essential intermediate steps. Such workflows are one of the central components of bioinformatics since the calculation must be parallel, reproducible and scalable.

The distribution of the detected methylation levels, e.g., values of several samples at a specific location, can often be described as a beta-mixture model. The standard approach for estimating the parameters for such a model, the EM algorithm, has problems for data points of 0 or 1, which are very common as methylation levels. For this reason, we have developed an alternative algorithm based on moments that overcome this disadvantage.

It is robust for data points within the closed interval $[0, 1]$ and can also be applied to similar data sets in addition to methylation levels.

This work deals not only with epigenetic but also with genetic variants. To analyze these, we present a second pipeline (`ape`) for data from targeted sequencing, where for example only genes are sequenced. The recognized variants then serve as input for our graphical environment `eagle`, a tool for computer scientists and geneticists to recognize possible causal genetic variants. As the name implies: The configuration of the analysis and presentation of the results is done via a graphical user interface. Unlike other tools, `eagle` is not based on databases, but on encapsulated hdf5 files. The use of this universal file-system-like data structure offers some advantages and makes the system easy to use especially for non-computer scientists.

At the end of the thesis, we use all methods presented for the detection, analysis, and characterization of interindividual DMRs between several donors. This leads to some computational challenges because DMR detection is usually performed on two different groups.

Our developed approach processes independent samples and calculates key metrics such as p-values and the number of undetectable DMRs. Through *whole genome association studies* (GWAS) on more than 1000 array data sets of methylation and variants, we show that (interindividual) DMRs as a subtype of epigenetics are related to genetic variation.

Acknowledgements

My last written words in this thesis are addressed to those who have helped and supported me over the past six years. First of all to my PhD supervisor Prof. Sven Rahmann. I am very grateful for the discussions on the whiteboard, his ideas, and solutions for my problems and the many things I learned from him. Then of course to Prof. Gunnar Klau, whom I would like to thank for agreeing immediately to become my second examiner, and to Prof. Jens Teubner and Prof. Johannes Fischer for their participation in the committee. My special thanks go to Prof. Bernhard Horsthemke, who was not only my mentor but also always a good cooperation partner in research and friend.

I would also like to thank my colleagues and former colleagues Marianna D'Addario, Dr. Daniela Beißer, Dr. Christina Czeschik, Corinna Ernst, Jan Forster, Elias Kuthe, Dr. Dominik Kopczynski, Dr. Johannes Köster, Dr. Marcel Martin, Felix Mölder, Henning Timm, Mareike Vogel, Marcel Wiesweg, Bianca Stöcker and Dr. Inken Wohlers as well as my former scientific colleagues Christopher Stahl and Sebastian Venier. Special thanks go to Till Hartmann for proofreading the dissertation. Many thanks for the exchange of ideas, the discussion and the fun we have.

I would like to thank not only my colleagues but also my current and former cooperation partners in human genetics. Dr. Tea Berulava, Dr. Jasmin Beygo, Alexander Kalmbach, Dr. Elsa Leitão, Dr. Katrin Rademacher, Dr. Petra Temming and Prof. Dagmar Wieczorek, without them none of my publications would have been possible. Special thanks go to Dr. Michael Zeschnigk, you and Prof. Bernhard Horsthemke have given me all my genetic knowledge.

Emotional support is just as necessary as technical support with knowledge. Mathias and Christin, many thanks that I can always come to you with problems and worries.

But my biggest thanks go to my beloved parents who were always there for me when I needed them, especially in difficult times. Without their support, I would not have made it.

Christopher Schröder
Essen, September 2018

List of Abbreviations

A	**A**denine
AF	**A**llele **F**requency
APE	**A** **P**ipeline for **E**xomes
bp	**b**ase **p**air
C	**C**ytosine
CAMEL	**C**alling and **A**nalysing **ME**thylation **L**evels
CpG	Desoxy**C**ytidin **p**hosphate Desoxy**G**uanosin
CGI	**C**pG **I**sland
CNV	**C**opy **N**umber **V**ariation
DEL	**DEL**etion
DL	**D**eep **L**earning
DNA	**D**eoxyribo**N**ucleic **A**cid
DMC	**D**ifferentially **M**ethylated **C**ytosine
DMR	**D**ifferentially **M**ethylated **R**egion
EAGLE	**E**xome **A**nalysis **G**raphica**L** **E**nvironment
G	**G**uanine
GWAS	**G**enome **W**ide **A**ssociation **S**tudy
GUI	**G**raphical **U**ser **I**nterface
INS	**INS**ertion
INV	**INV**ersion
INDEL	**INS**ertion or **DEL**etion
Kbp	**K**ilo **b**ase **p**air
LD	**L**inkage **D**esquilibrium
LP	**L**inear **P**rogramm
MAF	**M**inor **A**llele **F**requency
MAPQ	**MAP**ping **Q**uality
ML	**M**aximum **L**ikelehood
MLE	**M**aximum **L**ikelehood **E**stimation
NCP	**N**ucleosome **C**ore **P**article
NGS	**N**ext **G**eneration **S**equencing
NOMe-seq	**N**ucleosome **O**ccupancy **Me**thylome **seq**uencing
Mbp	**M**ega **b**ase **p**air
MDS	**M**inimum **D**ominating **S**et
PCR	**P**olymerase **C**hain **R**eaction
TRA	**T**ranslocation
TSS	**T**ranscription **S**tart **S**ide
SBS	**S**equencing **B**y **S**ynthesis
ssDNA	**s**single **s**tranded **D**eoxyribo**N**ucleic **A**cid
SV	**S**tructural **V**ariant
T	**T**hyime
U	**U**racil
WGBS	**W**hole **G**enome **B**isulphite **S**equencing
WGAS	**W**hole **G**enome **A**ssociation **S**tudy
WGS	**W**hole **G**enome **S**equencing

Contents

Chapter 1

Introduction

Several decades have passed since the four components of deoxyribonucleic acid (DNA) (Levene, 1919) and its structure were discovered (Watson and Crick, 1953). This molecule contains the genetic information responsible for the biological development of all known life forms. Since then, several techniques have been developed to identify DNA sequences and the genomes of many species have been fully sequenced, including the human genome (except for regions at the beginning and end of a chromosome and its center). As a result, and through constant further development and research in the fields of sequencing, sequence analysis, and human genetics, more and more disease-causing genetic variations have been identified. The ongoing reduction of sequencing costs enables additional studies of heritable phenotype changes that do not involve alterations in the DNA sequence, so-called *epigenetics* (Dupont et al., 2009).

One of the most important epigenetic mechanisms is methylation. Specific enzymes (DNA methyltransferases) catalyze a chemical reaction that replaces a single atom or group of atoms with a methyl group. Methylation typically takes place in the form of 5-methylcytosine on a cytosine followed directly by guanine (CpG). Figure 1.1 (b, c, and d) shows the structure of cytosine, 5-methylcytosine, and for comparison thymine. About 80% to 90% of all CpG sites in human DNA are methylated. In a biological context, CpG methylation serves as an inactivation marker or regulation for specific DNA regions (e.g., protein transcription, see Section 1.1) without changing the sequence itself.

While the vast majority of cells in an organism have the same DNA sequence, methylation differs in different cell types and may also change due to external influences or age (Berdasco and Esteller, 2011; Hackett and Surani, 2013; Heyn et al., 2012; Johnson et al., 2012). DNA methylation is involved in gene deactivation, tissue differentiation and diseases such as different types of cancer (Robertson, 2005; Suzuki and Bird, 2008). While the DNA sequence and its variations only differ from individual to individual and have been studied intensively, technological progress has only recently allowed detailed research into tissue-specific methylation. Neither most disease-related methylations nor the differences between most normal cell types have been investigated so far.

This thesis includes the basic principles of methylome sequencing as well as methods, workflows, and improvements of methylation processing step by step from data generation to downstream analysis of differentially methylated regions. It is divided into nine chapters. The chapter structure follows the process from the sequencing of the methylome to the final result of the differentially methylated regions and their annotation and downstream analysis.

Chapter 1 provides biological and technical knowledge about DNA, the genome, genes, variants, and sequencing. The information contained in this chapter is basic bioinformatics and is required for the rest of the thesis.

In Chapter 2 we develop a procedure for estimating the duplication rate of a sample library from a given subsample. The duplication rate is an essential indicator of sequencing data and a quality criterion for sequencing protocols. Our method can be used to estimate this metric for newly emerging protocols by cost-effectively sequencing only a small portion of the total data. Although the method is independent of the type of sequencing used, it is particularly useful for methylomes due to its high cost and large data volume.

Chapter 3 describes details of methylation data generation and bioinformatic processing, consisting of aligning raw data and identifying methylation levels by a process called methylation calling. We describe our developed method and its implementation in `Camel`, which has some advantages over existing tools. One aspect is the availability of an included downstream analysis of the called methylation data, for example, the detection of differently methylated regions (DMRs) between two sample groups, e.g., a comparison between tumor and blood methylation. Section 3.3 describes the implementation of our algorithm, which is based on a method of the widely used tool `bsmooth`. We present the original approach, our modifications, and evaluate our improvements by comparing the list of DMRs of both approaches.

For diploid genomes (see Section 1.1), the degree of methylation is typically either close to 1, close to 0, or close to 0.5 for allele-specific methylation. Some analysis, such as in Chapter 7, require the determination of the value distribution, which can be described by a mixture of beta distributions. In Chapter 4 we develop a method to adjust a mixture of beta distributions even at critical values close to 0.0 or 1.0, where a standard EM algorithm based on likelihoods would fail.

Chapter 5 shows two different workflows: one for processing methylomes and one for genetic variations, which mainly adresses data in selected regions such as the exome. The latter is also suitable for generating data for the system described in Chapter 6. Such workflows are core bioinformatics systems that allow to perform all steps from raw reads to the requested data (e.g., methylation levels) on a large scale automatically and are therefore an essential component in a study.

Sample-specific variations, such as SNPs (see Section 1.1), which are identified by the previously mentioned workflow then serve as input for further customer-specific downstream analyses. As mentioned above, a significant number of studies in recent years have dealt with disease-causing genetic variants. Also, sequencing costs have fallen to a level that enables medical tests based on sequencing. In Chapter 6 we present a system that allows geneticists to analyze exonic data via a web interface without the help of a bioinformatician. This system can also serve as a blueprint for methylation analysis as the continued increase in data production in this research area may require soon.

Finally, we use the results from Chapter 1-6 to perform a methylation analysis of real data in Chapter 7. In contrast to the usual two-group comparison, here we investigate into methylation differences between several individuals by adapting the methods presented. Besides, we show several smaller methods that we have developed for this purpose.

The work concludes in Chapter 8 with a summary and open problems, as well as possible solutions.

1.1 The Genome

The information and all images in this section are based on the book by Alberts et al. (2008) unless stated otherwise.

A genome carries the individual genetic information of an organism and is responsible for its development, functionality, growth, and proliferation. It is encoded by double helix structured macromolecules called deoxyribonucleic acid (DNA), which in turn is formed by two chains (strands) of four possible monomer units each, the so-called nucleotides. Each nucleotide consists of one of four nitrogen-containing nucleobases – cytosine (C), guanine (G), adenine (A) or thymine (T), a sugar called deoxyribose and a phosphate group. Covalent bonds between the sugar of one nucleotide and the phosphate of the next connect these nucleotides. Thus a DNA strand is *directed* with a free phosphate at one end (5′-end) and a free sugar at the other end (3′-end). Hydrogen bonds between the bases connect both strands of the double-stranded DNA molecule. Adenine binds (is complementary) to thymine and guanine to cytosine. Both strands are complementary to each other. This means that the base of the 3′-end of one strand is complementary to the 5′-end of the other strand. Computer scientists generally interpret the genome and DNA sequences as text over the alphabet A, C, G and T, which represents the four nucleobases. Figure 1.1 shows the chemical structure of a double-stranded DNA molecule consisting of four bases.

The genome in complex species such as plants and animals is organized in sets of double-stranded DNA molecules known as chromosomes. Most species have several copies of each chromosome in their chromosome set. The term *ploidy* refers to the number of copies of the chromosomes of an organism: monoploid (1 copy), diploid (2 copies), triploid (3 copies) and so on. There are 23 diploid chromosomes in the human genome. Therefore humans have two copies of each chromosome, with one exception: the sex chromosomes X and Y. While women have a diploid pair of X chromosomes, men have the Y chromosome replacing one of the two X chromosomes. A person inherits one chromosome (allele) from each parent, and at

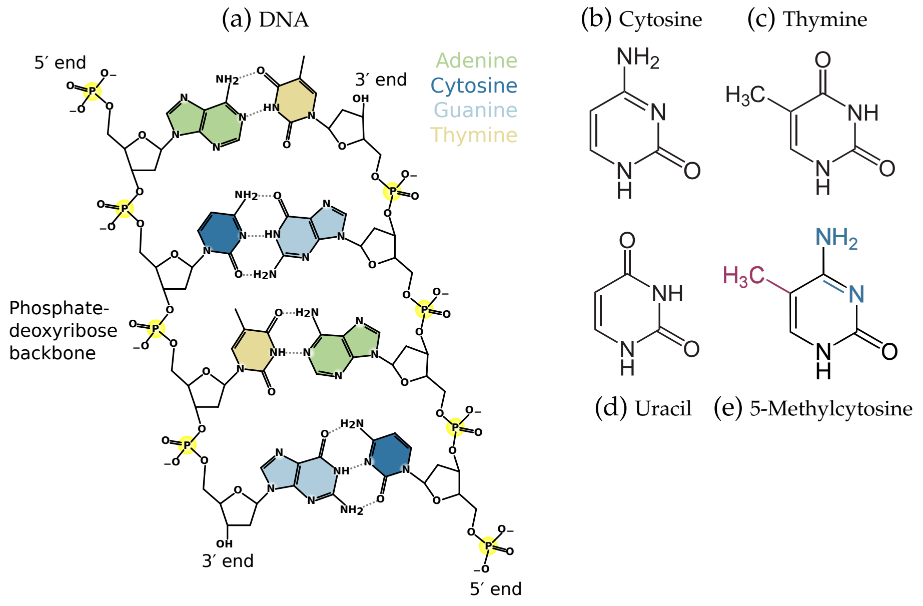


FIGURE 1.1: The chemical structure of (a) a four base pair long DNA molecule, (b) cytosine, (c) thymine, (d) uracil and (e) 5-methylcytosine with the difference between 5-methylcytosine and unmethylated cytosine (red) and the difference between thymine and 5-methylcytosine (blue).

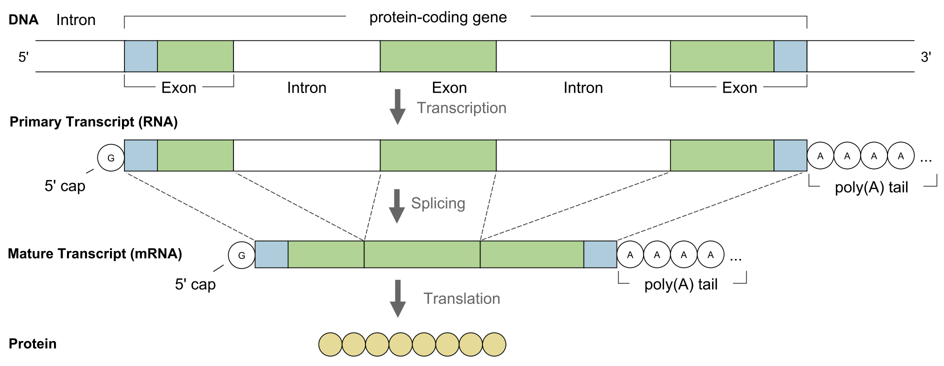


FIGURE 1.2: The process of protein biosynthesis. Special enzymes use the DNA nucleotide sequences of gene regions on the genome as a blueprint to generate protein amino sequences. The gene is transcribed into an RNA copy – the primary transcript. A process called *splicing* then builds the mature transcript by removing all intronic parts (white) from the RNA, and the remaining exons (green) are finally translated to synthesize the amino acid chain – the protein (yellow). So-called UTR regions (blue) flank protein-coding genes and the transcription adds a single guanine (5′ cap) to the 5′ UTR region and several adenines (poly(A)-tail) to the 3′ UTR region. We do not describe the functions of the UTR regions, 5′-Cap and Poly (A) at this point, they are not required in this thesis.

a very early stage of development, the DNA is exchanged blockwise (haploblock) between the two chromosomes. Often variants exist only on one allele (heterozygous variants), but sometimes also on both (homozygous variants).

Specific regions on the DNA called genes encode blueprints of proteins, chains of amino acids responsible for a variety of functions. While genes may have additional functions, in this work we focus on protein-coding genes. A gene itself consists of smaller subsequences called exons and introns (Figure 1.2), and the protein biosynthesis works as follows:

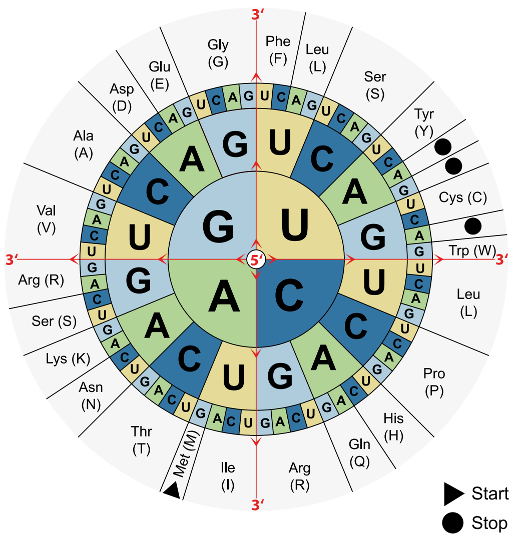
1. In the *transcription* phase, an RNA polymerase copies the DNA gene sequence into the precursor messenger RNA (pre-mRNA), but only one of the two strands is transcribed. This makes a gene strand-specific; the strand containing the gene is called *sense* strand in contrast to the complementary *antisense* strand. The resulting RNA is a copy of the original gene sequence with all exons and introns, but due to the nature of the RNA thymine is replaced by uracil (U), which has similar chemical properties to thymine (e.g., both T and U bind to A).

2. In a gene, only the exons carry the protein sequence information. To produce the protein, a process called RNA *splicing* cuts the pre-mRNA, removes introns and fuses the remaining exon sequences to form a new molecule called mature messenger RNA (mRNA).

3. The mRNA is transported from the nucleus into the cytoplasm to the ribosome, itself a protein complex. Here, a process called *translation* synthesizes the protein sequence consisting of 20 different types of amino acids from the information of the mRNA. This sequence is finally folded into the final protein.

In some cases, splicing removes not only the introns but also selective exonic sequences. Thus, a single gene can serve as a blueprint for numerous proteins, each with a different combination of exons. This so-called *alternative splicing* leads to different *transcripts* of a single gene. The sum of all genome-wide protein-coding sequences (exons) is called *exome*, which accounts for about 2% of human DNA. The remaining 98% builds the *non-coding DNA*.

During the translation phase at the ribosomes, triplets of mRNA nucleotides, so-called *codons*, are processed into an amino acid. This means that the coding part of the mRNA has a length of a multiple of three. Since each triplet consists of four different nucleotides, $4^3 = 64$ different combinations of {A,C,G,T} are possible for each codon. Three of these combinations (TAA, TAG, TGA) are *stop* codons and represent a stop signal that causes the end of translation. Another special case is the combination ATG, that represents the start of a translated sequence or the amino acid methionine (Met). Each of the remaining 60 codons encodes precisely one amino acid. However, since there are only 20 different proteins, some of the proteins have more than one codon coding for them. This redundancy is referred to as *codon-degeneration*. Figure 1.3 illustrates the coding scheme for all codons and amino acids. The upper part of the figure 1.4 shows a sequence of annotated codons and the resulting amino acid chain.



Name	Abbr.	#codons
Alanine	Ala	4
Arginine	Arg	6
Asparagine	Asn	2
Aspartic acid	Asp	2
Cysteine	Cys	2
Glutamic acid	Glu	2
Glutamine	Gln	2
Glycine	Gly	4
Histidine	His	2
Isoleucine	Ile	3
Leucine	Leu	6
Lysine	Lys	2
Methionine	Met	1
Phenylalanine	Phe	2
Proline	Pro	4
Serine	Ser	6
Threonine	Thr	4
Tryptophan	Trp	1
Tyrosine	Tyr	2
Valine	Val	4

FIGURE 1.3: Standard coding scheme (genetic code) of the various codons (nucleotide triplets). Read from the innermost (5′ end) to the outermost ring (3′ end), the codons are specified by the bases in this order. The outer grey ring indicates the amino acid that is formed. For example, codon UCC codes for the amino acid serine, codon CCG for proline and codon ACU for threonine. The figure is a modified version of the *coding sun* introduced by Bresch and Hausmann, 1972.

1.2 Variants

Two DNA sequences may differ in various ways. Usually, an individuals' sequence is compared to a previously generated reference genome. In the case of the human genome, the currently available reference is at version 38 (hg38), generated from anonymous volunteers by the Genome Reference Consortium (*Genome Reference Consortium*) and was released in December 2013. A difference in one or more bases of a sequence compared to a reference is called a variant and there exist two classes of variants:

A single nucleotide variant (SNV) is a variant where a single nucleotide differs from the given single reference base. The complete terminology is: Variant v substitutes base x by base y at chromosome c and position p (in sample s). Most scientific studies evaluating SNVs concentrate their efforts on these located in protein-coding regions (the *exome*) and the causing protein sequence change. Depending on the codon context, a variant affects the translated protein in different ways.

Synonymous variants change the base, but in a way such that the translated amino acid remains unaffected. This is possible due to the degeneracy of the genetic code. Figure 1.3 also shows a table of the 20 amino acids and the number of codons coding for them. One example for a synonymous variant might be a substitution of CCU to CCG. Although the last base of the codon changes, the resulting amino acid remains Proline. Even if synonymous mutations do not affect the amino acid, it has been shown that different synonymous SNVs may influence the protein folding process and

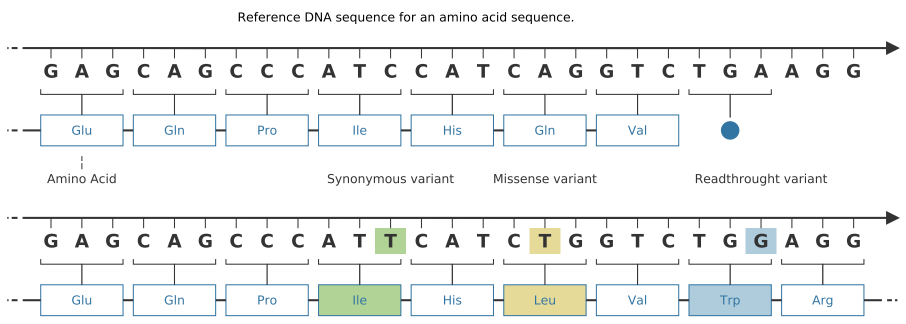

Reference DNA sequence for an amino acid sequence.
G A G C A G C C C A T C C A T C A G G T C T G A A G G
Glu
Gln
Pro
Ile
His
Gln
Val
Amino Acid
Synonymous variant
Missense variant
Readthrought variant
G A G C A G C C C A T T C A T C T G G T C T G G A G G
Glu
Gln
Pro
Ile
His
Leu
Val
Trp
Arg

FIGURE 1.4: Upper half: A reference DNA sequence and the corresponding amino acid chain that is translated from each triplet (codon) until a stop signal (blue circle) is reached. Lower half: The same sequence with three different modifications. ① a Synonym variant (green box) in which the codon with the modified nucleotide still codes for the same amino acid, ② a missense variant (yellow box) in which both the codon and its amino acid have changed and ③ a readthrough variant (blue boxes), which influences a stop codon and is replaced by a coding amino acid codon.

therefore change the protein structure (Sauna and Kimchi-Sarfaty, 2011).

Missense variants change the base of an affected codon so that the new codon is converted into another amino acid. This means that neither the affected codon nor the resulting codon is a stop codon and both codons are not synonymous. Protein structure and functionality can either be completely different, partially different or not changed at all compared to the original. Methods exist to estimate the effect of a variant on a protein structure – often referred to as *impact* (Ramensky et al., 2002; Schwarz et al., 2014; Wong and Zhang, 2014).

Nonsense variants change an amino acid coding codon to a stop codon. This causes the translation to stop before the original end is reached, truncating the amino acid sequence prematurely. This type of variant usually has a significant impact on protein functionality since several amino acids are missing.

Readthrough variants target a stop codon. The change in a stop codon causes prolongation of the translation process, resulting in more extended amino acid sequences. Codons are translated until reaching the next in-frame stop codon. In the case of an absent stop coding, the mRNA is called *nonstop mRNA*. A decrease by 100-fold in RNA expression due to mRNA degradation, translation repression, and protein destabilization was observed (Ito-Harashima et al., 2007).

The second category of variants are *INDELs* – an abbreviation for *insert and deletions* – where multiple bases are involved. Depending on the number of modified bases, this type usually has a much stronger impact on the protein structure.

Insertions are variants that have one or more base pairs inserted at specific locations compared to the reference. If inserts in coding regions and the number of inserted bases is a multiple of three, this results in one or more newly added codons. If the number of inserted bases is not a multiple of three, a so-called *frame shift* takes place, shifting the three-base reading frame for subsequent codons. Then the resulting amino acids and protein structures in the very most cases differ dramatically from the original reference.

Deletions are the opposite of insertions that lack one or more bases related to the reference. If the deletion occurs within an exon, multiple codons may be the target of the variation. Similar to inserts, a frameshift occurs when the number of deleted bases is not a multiple of three.

Duplications double a certain region of a chromosome. Often individual genes or groups of genes are duplicated as a whole.

Translocations are chromosomal abnormalities caused by the rearrangement of parts between non-homologous chromosomes. Chromosomal translocations are one of the most common types of genetic alterations and are molecular signatures for many cancers. They are considered the major causes of many diseases, especially lymphomas and leukemias (Nambiar and Raghavan, 2011).

The recognition of variants of a sample is the task of a so-called variant caller. Popular general variant callers are e.g. `GATK` (McKenna et al., 2010) and `Freeabyes` (Garrison and Marth, 2012), which we also use for our Exome pipeline `Ape` (see Section 5.2). There are also specialized tools like `Delly` (Rausch et al., 2012) for structure variants or `MuTect` (Cibulskis et al., 2013) for somatic variants between tumor and blood DNA.

In some cases, it may be useful or necessary to deal with variants or sequences containing positions whose nucleobases are variable rather than fixed. For this reason, the IUPAC nucleotide code was introduced, which allows several options to be offered for a single base position. Table 1.1 shows the complete IUPAC alphabet. For example, the letter Y in a DNA sequence represents the basic cytosine or thymine.

1.3 Next Generation Sequencing

Sequencing describes the process of identifying the nucleotide order of DNA or RNA molecules. Since 1975, most sequencing has been performed with the chain termination method (Sanger and Coulson, 1975; Sanger et al., 1977) proposed by Sanger, Nicklen and Coulson. Scientists were able to reconstruct the human genome and to create a reference sequence – a consensus sequence of several subjects. A new generation of sequencing methods was developed between 2005 and 2008, called *second* or *next generation sequencing* (NGS). Millions to billions of DNA strands are broken

IUPAC code	Mnemonic	Meaning	Complement
A	**A**denine	A	T
C	**C**ytosine	C	G
G	**G**uanine	G	C
T/U	**T**hymine/**U**racil	T/U	A
K	**K**eto	G or T	M
M	A**m**ino	A or C	K
R	Py**r**ine	A or G	Y
Y	P**y**rimidine	C or T	R
W	**W**eak	A or T	W
S	**S**trong	C or G	S
B	not A	C or G or T	V
D	not C	A or G or T	H
H	not G	A or C or T	D
V	not T/U	A or C or G	B
N	a**n**y base	A or C or T or G	N

TABLE 1.1: The IUPAC nucleotide code, a mnemonic to remember the meaning, the related bases and their complement.

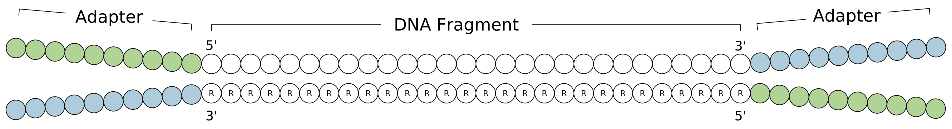


FIGURE 1.5: Sample preparation step of Illumina *sequencing by synthesis* (SBS). One of two different adapter sequences (blue and green) is attached to each side of the nucleotides (circles) of each strand and its reverse complement (R-marks).

down into shorter pieces, sequenced in parallel and then aligned in silico to the reference genome to identify their original chromosomal position. We have developed all methods in this work for the widely used (NGS) sequencing-by-synthesis (SBS) method developed by Illumina, which is responsible for generating more than 90% of the world's sequencing data in 2018 (Illumina Inc, 2017).

Sample preparation

The DNA is randomly cut into pieces of a particular target length, usually from about 100bp to 200bp (base pairs). These fragments are then extended at both ends by artificial adapter sequences, as shown in Figure 1.5. In contrast to the simplification shown in the figure, the adapter sequence consists of a primer and an index part, which are required for the simultaneous sequencing of several mixed samples. At this point, we do not provide further details about the adapters, as they are not required for this thesis. The double-stranded DNA is then denatured into two single-stranded DNA (ssDNA) fragments.

Cluster generation

The generated fragments are sequenced on a flow cell, a glass plate with billions of copies of two different short sequences (oligonucleotides or oligos). These oligos are reverse complementary to the previously attached adapter sequences of the ssDNA fragments. When applied to the flow cell, the adapter binds each fragment to its complementary oligos. A polymerase then synthesizes the chain of reverse complementary nucleic acids to complete the oligo and form a double-stranded DNA. This DNA is denatured into two single strands, and the now free original ssDNA is washed away. A process called bridge amplification copies the remaining backward complementary ssDNA molecules bound to the flow cell. Each strand bends and the free adapter hybridizes with the second type of oligo, which is then again complemented by a polymerase to double-stranded DNA. The DNA is denatured on two ssDNA strands, both of which are bound to the flow cell. This process (PCR cycle) is repeated several times in parallel for all strands of the flow cell. Finally, the inverted strands are cleaved and washed away so that clusters of identical forward DNA remain on the flow cell. Figure 1.6 shows the process of cluster generation.

Sequencing by synthesis

The final step of sequencing consists of biochemical treatment and image-based data acquisition. A short ssDNA sequence (primer) connects to the adapter and is extended step by step with complementary nucleotides. Each of the nucleotides A, C, G, T has a different fluorescent tag. When a nucleotide binds to a (complementary) base, the tag is released after each cycle, and all free tags are measured in parallel. The detected wavelength then determines the bound base for each cluster. Figure 1.7 shows the sequencing process. After a predefined number of sequencing cycles, the product is denatured and washed away. Again, a single bridge amplification step is applied, and now the forward strands are washed away similar to the cluster generation shown in Figure 1.6 steps 4 and 5. The sequencing by synthesis step is repeated for the reverse strand, and the resulting data is computationally analyzed and output as two sequences (read pair) and a quality score for each base (see Chapter 1.6). Today's sequencing techniques produce several million reads per sample with a typical length of 50bp-150bp.

Information about SBS and the figures are based on Illumina Inc (2017).

1.4 Sample preparation modification

Depending on the desired information, the various sample preparation steps can be extended or modified in different ways. The following list gives an overview of the main techniques mentioned in this thesis.

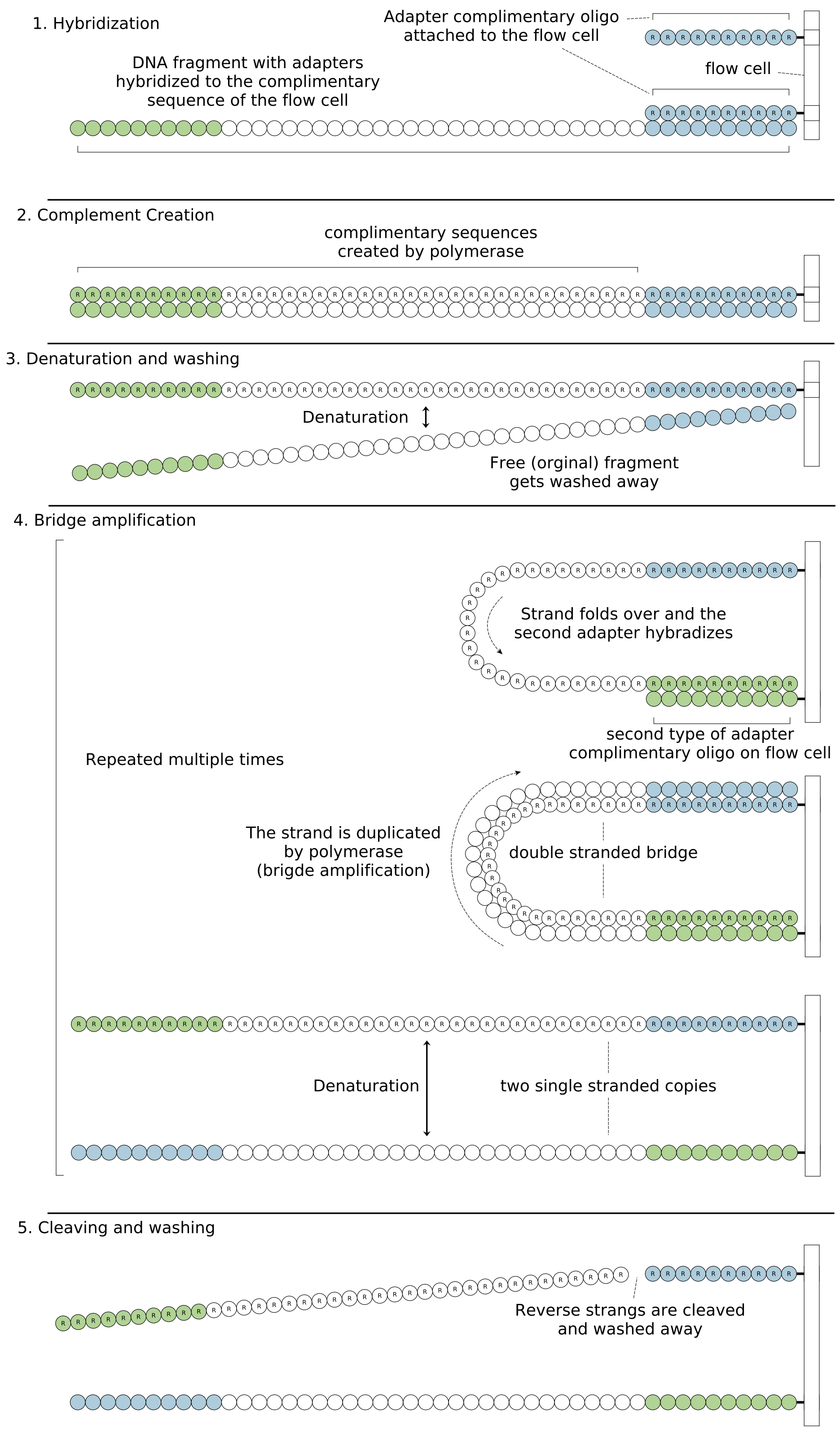

1. Hybridization
Adapter complimentary oligo attached to the flow cell
DNA fragment with adapters hybridized to the complimentary sequence of the flow cell
flow cell
2. Complement Creation
complimentary sequences created by polymerase
3. Denaturation and washing
Denaturation
Free (orginal) fragment gets washed away
4. Bridge amplification
Strand folds over and the second adapter hybradizes
second type of adapter complimentary oligo on flow cell
Repeated multiple times
The strand is duplicated by polymerase (brigde amplification)
double stranded bridge
Denaturation
two single stranded copies
5. Cleaving and washing
Reverse strangs are cleaved and washed away
R

FIGURE 1.6: Cluster generation steps of Illumina sequencing by synthesis (SBS).

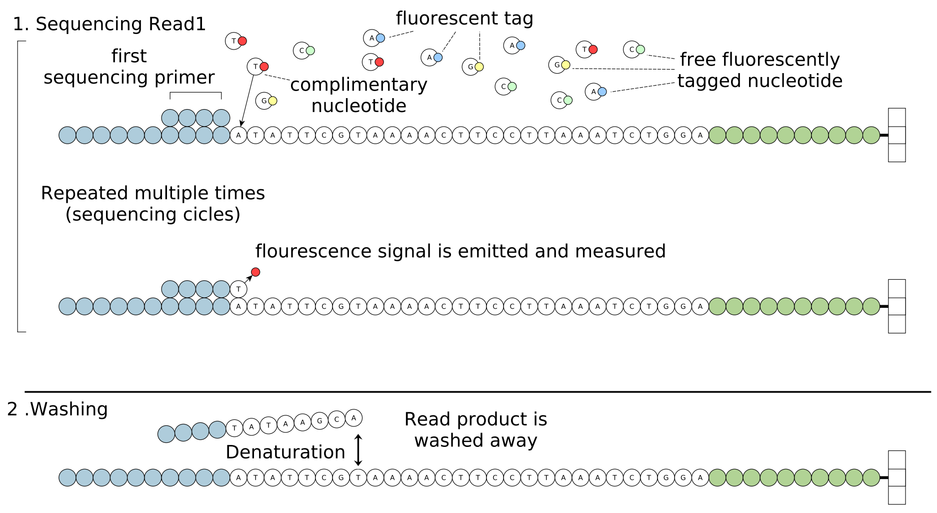

1. Sequencing Read1
first sequencing primer
fluorescent tag
complimentary nucleotide
free fluorescently tagged nucleotide
Repeated multiple times (sequencing cicles)
flourescence signal is emitted and measured
2 .Washing
Denaturation
Read product is washed away

FIGURE 1.7: The sequencing step of Illumina sequencing by synthesis (SBS).

Whole Genome Sequencing (WGS) uses fragments of the entire genome-wide DNA. This method is expensive and the most general of all techniques. It generates the genetic information of the entire genome and can be used to identify point mutations, small and large structural variations, and copy number variations.

Whole Exome Sequencing (WES) targets exonic regions. Only 2% of the human DNA sequence is used as protein blueprints (see Section 1.1). Modifications in these regions are of great interest because a variation can directly alter their translated protein structure and thus affect protein functionality. It is possible to extract only read from predefined coding regions with commercially available capture kits using hybridization techniques and biotinylated oligonucleotide probes complementary to the target region. Such a capture kit can also capture regions other than the exome, which is generally summarized under the term *targeted sequencing*.

Whole Genome Bisulphite Sequencing (WGBS) is used for genome-wide measurement of DNA methylation at single base pair resolution. During library preparation, the DNA is treated with sodium bisulfite, which converts the unmethylated cytosine in the CpG context to uracil (U) and leaves 5-methylcytosine unaffected. Uracil is then recognized as thymine by the sequencer. This work mainly deals with this kind of sequencing, and the technique is described in detail in Chapter 3.

Nucleosome allocation and methylome sequencing (NOMe-seq) is a modification of bisulfite sequencing using a GpC methyltransferase to methylate cytosine in GpC instead of in the CpG context (Kelly et al., 2012). In general, most of a DNA strand is wrapped around several scaffold proteins (histone nuclei), while about 146bp of the DNA (nucleosome nucleus particles, NCP) is wrapped around the histone octamer, each consisting of 2 copies of the nuclear histones (Felsenfeld and Groudine, 2003). The complex of histone nucleus and NCP is called *nucleosome*. When proteins

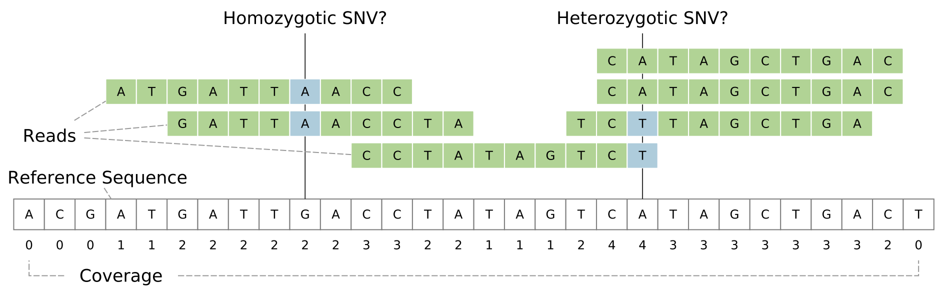


FIGURE 1.8: Multiple DNA reads (green) of length ten at their genomic position and the reference sequence (white squares) with consensus base at position p. The coverage (lower numbers) counts the number of bases from all reads overlapping p and alternative bases (blue) relative to the reference base show either a variant or a sequencing error.

are translated, this structure changes in such a way that RNA polymerases can access the DNA. The added methyltransferase only comes into contact with GpC at these accessible sites, the so-called open chromatin. GpCs are never methylated in mammalian genomes, and methylated cytosine implies open chromatin in this context. Therefore, this method allows the analysis of open chromatin states.

1.5 Alignment and coverage

The extraction of genetic or epigenetic information often requires knowledge of the exact origin positions of the generated reads. Instead of assembling the entirety of all reads into a sequence, the sequences are usually aligned against the species-specific reference genome by programs called *read mappers*. This task is performed by tools such as the widely used `BWA` (Li and Durbin, 2009) or `bowtie2` (Langmead et al., 2009) done. In addition to the genomic position of each pair, differences to the reference and mismapping probability are also determined.

The sequencing generates from several nanograms of DNA from many cells (except single-cell sequencing) millions to billions of read pairs, which usually overlap. The number of overlapping reads of a single position p is called *coverage* at p. Figure 1.8 shows overlapping reads at a genomic position and coverage. Base differences in comparision to the reference can be either the result of a technical error or an existing variant at this position. In order to distinguish these two cases and not to miss real variants, high coverage is one of the crucial quality metrics in genome sequencing. Algorithms of so-called SNV callers are usually able to identify errors if the coverage is high enough. The sequencing costs are almost proportional to the desired coverage.

Q	P	Accuracy (1-P)	Usage
0	N/A	N/A	special case, e.g. read unmapped
10	0.1	90%	minimum threshold for SNV analysis
13	0.05	95%	
20	0.01	99%	
30	0.001	99.9%	
40	0.0001	99.99%	
50	0.00001	99.999%	typical threshold for SNP analysis
60	0.000001	99.9999%	maximum read mapping quality

TABLE 1.2: Example quality to error probability transformations and their usage.

1.6 Quality scores

Bioinformatic analyses often deal with tiny probabilities in the calculation of uncertainty; floating point numbers with large negative exponents. For subsequent analyses, it is often sufficient to consider the size of the exponent rather than dealing with exact values. For this reason, the so-called *Phred* scaling was established (Ewing et al., 1998), which converts a probability P into a quality score Q by

$$Q = -10 \log_{10} P$$

and equivalently Q to P by

$$P = 10^{-Q/10}.$$

The Phred scaled quality score is used in a variety of bioinformatics applications and is typically stored as an integer in a human-readable range of $[0, 255]$ or $[0, 65535]$. Although the quality score can always be used instead of small probabilities, the values in table 1.2 are of particular importance for this work.

Base Quality Scores (BQS) are generated by the sequencing platform for each sequenced base. It reflects the Phred scaled probability that the base was incorrectly determined, where $BQS \in [0, 60]$ (Ewing and Green, 1998; Pavlopoulos et al., 2013)..

Mapping Quality Scores (MAPQ/MQS) are a measure of confidence in the correctness of an identified genomic position. It is estimated by read aligners and was introduced in 2008 by Heng Li and Richard Durbin (Li et al., 2008). Mapping qualities are in the range of $[0, 60]$ and depend on the null model of the algorithm used, while MAPQ $= 0$ is usually assigned to reads with several equivalent positions (ambiguous reads) or no mapping position at all.

SNP quality is the measure of uncertainty for an estimated variant genotype. They are calculated by a variant caller and calculated from the base call, orientation, and assembly and coupled with prior information such as allele frequencies and patterns of relationship imbalances (LD) (Nielsen et al., 2011).

1.7 Hdf5

Most of our tools presented in this thesis store and load their data in *Hierarchical Data Format Version 5* (hdf5) – an universal data format for storing and organizing large amounts of data. Similar to directories and files, hdf5 files are organized in a hierarchical structure with groups and records. A group contains any number of additional groups or data sets and optional metadata. A data set is a multidimensional data array (and optional metadata) that can consist of primitive data types (whole or unsigned integers of 8, 16, 32 or 64 bits, floating point numbers of 32 or 64 bits, references or strings) or structured data consisting of multiple heterozygous data of primitive types.

Hdf5 supports compression and chunked storage, and also stores an index in the form of B-trees in the hdf5 file itself, allowing random access to groups and records. There are hdf5 parsing libraries for all common languages, such as h5py for python (Collette, 2008), rhdf5 for R (Fischer and Pau, 2016) or hfd5.jl for julia (Holy, 2016).

Chapter 2

Duplicate Rate Estimation

Current NGS mass technologies require several PCR amplification steps. Depending on the number of PCR cycles, thousands of copies of fragments are generated, and multiple copies of the same fragment can be sequenced multiple times. Such PCR duplicates are usually identified during data processing and marked as such in order to avoid unintentional biases. It does not make sense to sequence a library with many duplicates in great depth: Most reads will then be duplicates of each other and excluded from subsequent analysis. Many different factors influence the proportion of PCR duplicates (duplicate rate) such as the amount of DNA used and the preparation of the library. If possible, these factors should be optimized by several iterations in order to minimize the number of duplicates.

A strategy of some projects is to first sequence a small sub-sample of a library, for example in the range of 1% to 10%, to determine the duplicate rate and then extrapolate the duplicate rate of the entire library. Therefore, methods for estimating the duplicate rate at increased sequencing depth are of great interest for such large projects or implementations and parameter tuning of new sequence protocols. The equivalent question is that of *library complexity*, which refers to the number of observable original reads (counting each fragment present in multiple copies only once) at a given sequencing depth.

In this chapter, we propose an explicit optimization approach based on elementary probability theory and combinatorics. We formulate the complexity estimation problem of the library first as a convex optimization problem and then give a more practical linear program to avoid the inconsistency of the original question. This chapter is based on a lecture at the German Conference for Bioinformatics (GCB) 2015 and its proceedings (Schröder and Rahmann, 2015), where Sven Rahmann wrote most of the text.

2.1 Family Partitioning

The duplicate rate of a dataset is usually estimated as follows. DNA fragments are aligned with a read mapper and then compared based on their identified genomic position (and fragment length, if applicable). For paired-end reads, the mapping positions of both ends are taken into account. The reason for this approach is that it is highly unlikely that independently generated DNA fragments at both ends have the same starting position in the genome with typical covers. This approach is problematic if only short single-end reads are available or if the coverage is extremely high. In this case, it is impossible to distinguish independent fragments from PCR duplicates, and the approach will overestimate the actual duplicate rate. Another approach is to estimate the duplicate rate by sequence identity (modulo sequencing errors). The use of the exact identity is below the true duplicate rate, and this option is preferable only if there is no reference genome on which the reads can be mapped.

Notwithstanding these problems, there are working operational definitions of the duplicate rate and tools that mark and count duplicates, such as `picard-tools MarkDuplicates`. (Broad Institute, 2013) and `samtools rmdup`. (Li et al., 2009). For our purposes, the exact definition of duplicates is of secondary importance. We assume that we have a large set of sequential and potentially mapped single-end or paired-end reads and a procedure that can identify reads as duplicates of each other. This can be achieved either by comparing sequence contents or by mapping information.

Thus the set of fragments can be partitioned into *families*, such that each family contains all copies of a fragment and no other fragments. We say that a family is of *type* c or a *type-c family* if it contains exactly c elements, one of which is *original* and $c-1$ are *duplicates*. The number a_c of type-c families in the complete dataset is called the *occupancy* number of c. The number of fragments in type-c families is thus $c \cdot a_c$. The vector $a = (a_1, a_2, \dots)$ is called the *occupancy vector* of the dataset.

Thus, the total number of sequenced fragments is

$$N = \sum_{c \geq 1} c \cdot a_c \,.$$

The *complexity* Γ of the dataset is the number of original fragments, i.e., the total number of families:

$$\Gamma = \sum_{c \geq 1} a_c. \tag{2.1}$$

The *duplicate rate* of the full dataset then is

$$d = (N - \Gamma)/N = \frac{1}{N} \sum_{c \geq 1} (c-1) \cdot a_c \,. \tag{2.2}$$

2.2 The Subsampling Process

When taking a random subsample of size $n \ll N$, a family of type c will have $0 \leq k \leq c$ of its c members in the subsample. We call a family with k members in the subsample a family of *subtype* k regardless of its type c.

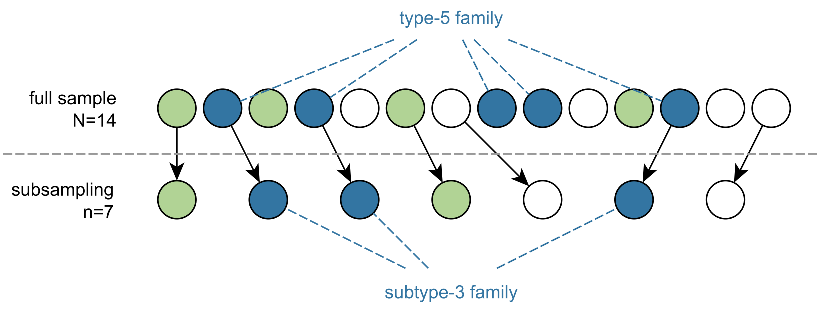


FIGURE 2.1: Example of the subsampling process of a type-5 family to a subtype-3 family (blue). The example contains also a type-4 (green) and a type-5 (white) family, both sampled to a subtype-2 family.

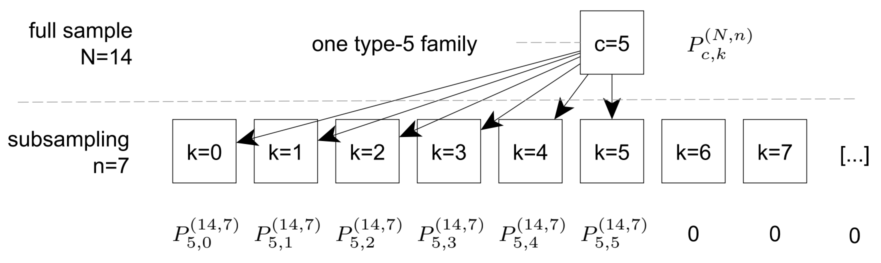


FIGURE 2.2: Example of the downsampling process of a single type-5 family and illustration of the probabilities $P_{5,k}^{(N,n)}$ to obtain a specific subtype k.

Figure 2.1 shows an example of sub-sampling a sample of 14 elements to a sub-sample of size seven. The full sample contains a type-5 family that is reduced in the subsample to a subtype-3 family represented as blue balls. In this example, the occupancy vector for the entire example is given by $a = (0, 0, 0, 0, 0, 1, 2)$. There is exactly *one* family with four members (green) and *two* families with five members (white and blue).

We can explicitly calculate the probability that a type c family will be down-sampled to subtype k for each $c \geq 1$ and each $k \geq 0$.

The probability that a family of type c is of subtype k for sample size N and subsample size n is given by the hypergeometric probability

$$P_{c,k}^{(N,n)} = \binom{c}{k}\binom{N-c}{n-k} \Big/ \binom{N}{n} \qquad (k = 0, \ldots, c).$$

The reason is a standard combinatorial argument: There are $\binom{N}{n}$ possible size-n subsamples of a set of size N, there are $\binom{c}{k}$ ways to choose k objects from c, representing the copies of the particular read, and $\binom{N-c}{n-k}$ ways to choose the remaining objects. Figure 2.2 illustrates probabilities of a single type-5 family to become a specific subtype-k family.

Note that families of subtype $k = 0$ are not observed, while families of subtype $k \geq 1$ are observed, but it is unknown from which type c they

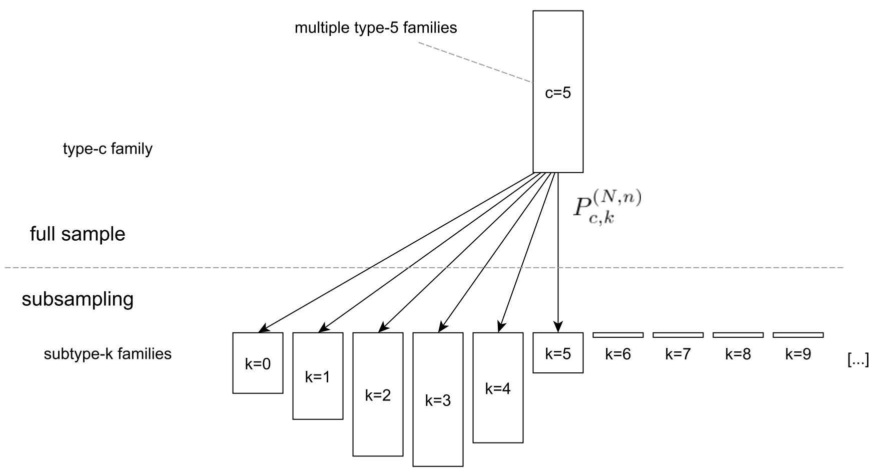


FIGURE 2.3: Example of the downsampling process of multiple type-5 families. The height of each rectangle represents the number of family members for each type or subtype.

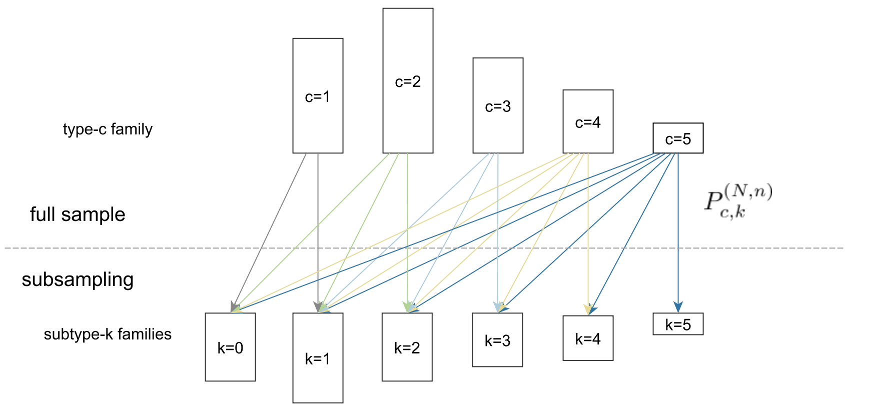


FIGURE 2.4: Example of families of type 1 to 5 downsampled to subtype k families. The height of each rectangle represents the number of members in each family.

originate.

For a given occupancy vector $a = (a_1, \ldots, a_N)$ of the complete dataset, we can thus compute the *expected occupancy vector* of the subsample as $x = x(a) = (x_0, \ldots, x_n)$ with

$$x_k = x_k(a) = \sum_{c \geq 1} a_c \cdot P_{c,k}^{(N,n)} \qquad (0 \leq k \leq n). \tag{2.3}$$

Figure 2.4 shows an example of subfamilies, where the number of members is such a linear combination of the downsampling process of multiple families.

2.3 The Estimation Problem

Given the *observed occupancy vector* $o = (o_k)_{k \geq 1}$ of the subsample consisting of $n = \sum_{k \geq 1} k \cdot o_k$ fragments, and given the size N of the full dataset, estimate the duplicate rate d (or equivalently the complexity Γ) of the full dataset.

The presentation so far suggests that the problem can be addressed by estimating the occupancy vector $a = (a_c)$ of the full dataset such that the expected occupancy vector $x = (x_k)$ of the subsample given by Equation (2.3) is close to the observed occupancy vector $o = (o_k)$. It should be noted that the precise values of (a_c) are not asked for, but estimating their sum Γ is sufficient for solving the problem. The *classical* solution (Good and Toulmin, 1956) approximates the subsampling process by a Poisson process that is then extrapolated for $N/n \leq 2$. Convergence acceleration methods of power series can be used to extend this approach up to $N/n \leq 5$. For larger factors, the variance of the estimator becomes too large to provide useful results. Precisely for this reason, Daley and Smith, 2013 has developed a robust extrapolation of this approach for larger N/n, based on rational functions.

In this work, we show a direct and explicit optimization approach for the problem. Indeed, we translate the requirement that $x = (x_k) \approx o = (o_k)$ into an objective function and model the constraints directly. Our approach makes it possible to assess the uncertainty of the estimate and determine confidence intervals.

2.4 Modeling the Optimization Problem

Before stating the optimization problem, a few remarks are in order.

1. There are several possibilities to measure the closeness between x and o, which is discussed below.
2. The value of x_0, i.e., the expected number of families that are never observed in the subsample, cannot be related to an observation and does not take part in the objective function.
3. While N and n can be as large as several hundred million and are the maximally possible lengths of a and o, respectively, there are no fragments with excessively many copies in typical real datasets. Thus the observed nonzero entries (o_k) typically end at a small value K_{last}, i.e. $o_k = 0$ for $k > K_{\text{last}}$. Because families of large type c would generate families of large subtype $k > K_{\text{last}}$, we can effectively put an upper bound C on the read types to consider and restrict the occupancy vector a to $(a_1, \ldots, a_C)$. Likewise, this choice of C bounds the subtype of the reads that can be reasonably generated by subsampling, and we limit the occupancy vector x to $(x_1, \ldots, x_K)$. It follows that only

the hypergeometric probabilities $P_{c,k}^{(N,n)}$ are needed for $0 \leq c \leq C$, $0 \leq k \leq K$. The choice of C and K is discussed in Section 2.4.4.

4. While the true occupancy vector of the complete sample consists of integer counts, we shall not constrain the estimates a_c to integers. Solving an integer-constrained problem is much more difficult, and our goal is not the precise estimation of the single a_c values, but an approximate estimate of $\Gamma = \sum_c a_c$.

The following shows how to quantify the difference between the subsample's expected occupancy vector $x = x(a) = (x_1, \ldots, x_K)$ and the observation $o = (o_1, \ldots, o_{K_{\text{last}}}, 0, \ldots, 0)$ with $K \geq K_{\text{last}}$ elements.

2.4.1 Maximum Likelihood Approximation

Consider the random variable X_k that counts families of subtype k after subsampling. Such families arise from type-c families for any $c \geq k$, and the *success probability* for a type-c family to become of subtype k is precisely $P_{c,k}^{(N,n)}$. Thus we can write $X_k = \sum_{c \geq k} X_{k,c}$, where $X_{k,c}$ has a Binomial distribution with size parameter a_c and success probability $P_{c,k}^{(N,n)}$. Under mild assumptions, this can be approximated by a Poisson distribution with parameter $\lambda_{k,c} := a_c \cdot P_{c,k}^{(N,n)}$. Assuming approximate independence of the Poisson-distributed $X_{k,c}$ (the only violation is the known size n of the subsample) we can approximate their sum X_k by a Poisson distribution with parameter $\sum_c \lambda_{k,c} = \sum_c a_c \cdot P_{c,k}^{(N,n)} = x_k$ according to Eq. (2.3). For large x_k, this is approximately a Normal distribution with expectation x_k and variance x_k. (In fact, because of the known size, the $X_{k,c}$ have a small negative correlation, and the overall variance is slightly smaller than x_k.) With this approximation in mind, the (approximate) maximum likelihood approach suggests minimizing

$$\begin{aligned} f_1(a) &:= \sum_{k=1}^{K} (x_k(a) - o_k)^2 / x_k(a) \\ &= \sum_{k=1}^{K} (x_k(a) + o_k^2 / x_k(a)) + \text{const.} \end{aligned} \tag{2.4}$$

It can be shown that f_1 is convex by proofing the convexity of

$$x \mapsto \sum_{k=1}^{K} (x_k + c_k / x_k), \tag{2.5}$$

as $x(a)$ is a linear function of a.

A function $f(x)$ is convex if the following inequality is true:

$$f(\lambda x_1 + (1-\lambda)x_2) \leq \lambda f(x_1) + (1-\lambda) f(x_2) \quad \text{for} \quad \lambda \in [0,1] \tag{2.6}$$

Theorem 1. *The sum of two convex functions $h = f + g$ is again convex:*

$$\begin{aligned}
&h(\lambda \cdot x_1 + (1-\lambda) \cdot x_2) \\
&\quad = (g+f)(\lambda \cdot x_1 + (1-\lambda) \cdot x_2) \\
&\quad = f(\lambda \cdot x_1 + (1-\lambda) \cdot x_2) + g(\lambda \cdot x_1 + (1-\lambda) \cdot x_2) \\
&\leq \lambda \cdot f(x_1) + (1-\lambda) \cdot f(x_2) + \lambda \cdot g(x_1) + (1-\lambda) \cdot g(x_2) \\
&\quad = \lambda \cdot (f(x_1) + g(x_1)) + (1-\lambda) \cdot (f(x_1) + g(x_2)) \\
&\quad = \lambda \cdot (f+g)(x_1) + (1-\lambda) \cdot (f+g)(x_2) \\
&\quad = \lambda \cdot h(x_1) + (1-\lambda) \cdot h(x_2) \qquad \square
\end{aligned}$$

Equation 2.5 can be transformed into

$$x \mapsto \sum_{k=1}^{K} x_k + \sum_{k=1}^{K} c_k / x_k \tag{2.7}$$

and Theorem 1 allows us to infer convexity when both sum terms are convex, and to infer convexity of a single sum when all summands are convex. The coordinate selection $x \mapsto x_k$ is convex (linear) for each $k \in 1..K$. Term c/x is convex for $x > 0$, which is proofen by the use of inequation 2.6:

$$\begin{aligned}
\frac{c}{\lambda x_1 + (1-\lambda)x_2} &\leq \lambda \frac{c}{x_1} + (1-\lambda)\frac{c}{x_2} && \Leftrightarrow \\
\frac{1}{\lambda x_1 + (1-\lambda)x_2} &\leq \frac{\lambda}{x_1} + \frac{1-\lambda}{x_2} && \Leftrightarrow \\
\frac{1}{\lambda(x_1 - x_2) + x_2} &\leq \frac{\lambda x_2 + (1-\lambda)x_1}{x_1 x_2} && \Leftrightarrow \\
\frac{1}{\lambda(x_1 - x_2) + x_2} &\leq \frac{\lambda(x_2 - x_1) + x_1}{x_1 x_2} && \Leftrightarrow \\
x_1 x_2 &\leq (\lambda(x_1 - x_2) + x_2)(\lambda(x_2 - x_1) + x_1) && \Leftrightarrow \\
0 &\leq (\lambda(x_1 - x_2) + x_2)(\lambda(x_2 - x_1) + x_1) - x_1 x_2 && \Leftrightarrow \\
0 &\leq -\lambda^2(x_1 - x_2)^2 + \lambda x_1(x_1 - x_2) - \lambda x_2(x_1 - x_2) && \Leftrightarrow \\
0 &\leq -\lambda^2(x_1 - x_2)^2 + \lambda(x_1 - x_2)(x_1 - x_2) && \Leftrightarrow \\
0 &\leq -\lambda^2(x_1 - x_2)^2 + \lambda(x_1 - x_2)^2 && \Leftrightarrow \\
0 &\leq (\lambda - \lambda^2)(x_1 - x_2)^2 && \Leftrightarrow \\
0 &\leq \lambda(1-\lambda)(x_1 - x_2)^2 && \square
\end{aligned}$$

since $\lambda(1-\lambda) \geq 0$ and $(x_1 - x_2)^2 > 0$ for $\lambda \in [0;1]$ and $x_1, x_2 > 0$.

As all summands of both sums of Equation 2.7 are convex, both sums are convex and therefore the function itself.

While f_1 would be the most principled objective function to use, several properties of the problem make f_1 both inconvenient and unnecessary to use in practice.

1. While there has been much progress on convex optimization methods, they are not as easy to use, numerically robust or widely available as linear optimization methods.

2. We found that in particular, the numerical robustness issues prevent us from successfully using the existing libraries such as CVXOPT (Andersen et al., 2003). Realistic numbers of involved fragments are approximately $n = 10^6$ and $N = 10^8$, while some of the small but still relevant hypergeometric probabilities are 10^{-9}.

3. Because of the compression effects of subsampling (we are trying to estimate an occupancy vector a with C elements from a smaller occupancy vector o with $K \ll C$ elements), it will in most cases be no problem to fit $x(a)$ to o almost exactly with several choices of a, and the optimal objective function will be close to zero.

2.4.2 Linear Program

The above considerations suggest using a linear proxy for f_1. We replace each term by its square root, and as $x_k \approx o_k$, we replace the x_k in the denominator by $o_k + 1$ to avoid singularities at zero. The objective function

$$f(a) := \sum_{k=1}^{K} |x_k(a) - o_k| \,/\, \sqrt{o_k + 1} \tag{2.8}$$

is obtained. Each term approximately specifies how many standard deviations o_k differs from its expected value $x_k(a)$ for a candidate a (as long as $o_k \approx x_k(a)$, which we can expect from the solution given the above considerations).

Using f, the problem can be cast as a linear program (LP) with variables $(a_1, \dots, a_C, x_1, \dots, x_K)$ and weights $w_k := 1/\sqrt{o_k + 1}$ by transforming the absolute value term into linear inequalities using additional auxiliary variables $(\delta_1, \dots, \delta_K)$ with $\delta_k \geq |o_k - x_k|$:

$$\text{minimize} \sum_{k=1}^{K} w_k \cdot \delta_k \qquad \text{such that} \tag{2.9}$$

$$o_k - x_k \leq \delta_k \qquad (k = 1, \dots, K), \tag{2.10}$$

$$-o_k + x_k \leq \delta_k \qquad (k = 1, \dots, K), \tag{2.11}$$

$$x_k = \sum_{c=k}^{C} a_c \cdot P_{c,k}^{(N,n)} \qquad (k = 1, \dots, K), \tag{2.12}$$

$$N = \sum_{c=1}^{C} c \cdot a_c\,, \tag{2.13}$$

$$a_c \geq 0 \qquad (c = 1, \dots, C)\,. \tag{2.14}$$

Some observations on the LP follow.

1. The above constraints imply additional ones. For example, using that the expectation of the hypergeometric distribution $P^{(N,n)}_{c,\cdot}$ is $(cn)/N$, together with (2.12) and (2.13), shows that

$$\sum_k k\, x_k = \sum_{c\geq 1} a_c \cdot \sum_{k\geq 1} k\, P^{(N,n)}_{c,k} = \sum_{c\geq 1} a_c \cdot (cn)/N = n,$$

   i.e. the total number of fragments in x is constrained to be the same as that in o.

   It also follows that $x_k \geq 0$ for all k, as well as $a_c \leq N/c$ for all c and $x_k \leq n/k$ for all k.

2. The variables (x_k) can be eliminated from the problem by replacing x_k with the appropriate linear combination of a given by (2.12). The LP formulation here includes them for readability.

3. Solving the LP (2.9)–(2.14) will give one (out of possibly many almost equivalent) solutions for a, from which we derive an estimate of the complexity Γ or of the duplicate rate d using Eqs. (2.1) and (2.2).

4. In an optimal solution (a^*, x^*, δ^*), the k-th term $w_k\, \delta^*_k = \delta^*_k/\sqrt{o_k + 1}$ states approximately how many standard deviations the observed value o_k deviates from the expected x^*_k for the computed a^*. For subtypes k with sufficiently many observations, say ≥ 25, we may expect small values (less than 1).

2.4.3 Approximate Confidence Intervals

As already stated, the problem of estimating a from o is ill-conditioned. For large enough c, two adjacent rows of the hypergeometric probability matrix, $P^{(N,n)}_{c,\cdot}$ and $P^{(N,n)}_{c+1,\cdot}$ have very similar entries, and for an individual observed read of subtype k, the type c is undecidable. There may be solutions $a' \neq a$ that achieve almost the same objective function value, i.e. $x(a') \approx x(a)$ even though $a' - a$ is large by any measure. Thus an interval estimate $[d_{\text{min}}, d_{\text{max}}]$ may be more informative than a point estimate of d.

We proceed as follows. Given the optimal solution (a^*, x^*, δ^*) of the linear program (2.9)–(2.14), we set up a new linear program to minimize resp. maximize the duplicate rate (maximize resp. minimize the complexity $\Gamma = \sum_{c=1}^{C} a_c$) under constraints that the admissible solutions do not differ much from the optimal one. In particular, we constrain each term of the original objective function to stay within two standard deviations unless it was already larger in the original solution,

$$w_k \cdot \delta_k \leq \max\{2, w_k \cdot \delta^*_k\} \qquad (k = 1, \dots, K). \tag{2.15}$$

Additionally, we constrain the sum of these terms by

$$\sum_{k\in K_!} w_k \cdot \delta_k \leq \sum_{k\in K_!} \max\{Z, w_k \cdot \delta^*_k\}, \tag{2.16}$$

where $K_! := \{k \mid o_k \geq 25\}$ is the set of ks with a non-negligible amount of observations and $Z > 0$ is a tunable parameter that we set to $Z := 1$ by default. We empirically found this choice to produce approximate 95% confidence intervals. In summary, the two LPs to be solved are

$$\text{minimize / maximize} \sum_{c=1}^{C} a_c, \tag{2.17}$$

such that (2.10)–(2.16) hold. This produces solutions $a^{\min}$ and $a^{\max}$, from which we derive the duplicate rates $d^{\max}$ and $d^{\min}$, respectively, by (2.2).

As a remark, we have chosen this particular approach for numerical robustness. While one could argue that a more natural approach is to require that the total variance $V := \sum_{k=1}^{K} w_k^2 \delta_k^2$ be limited by $4K$, this would be the solution of a quadratically constrained program. Our attempts with freely available solvers led to severe numerical problems, no solutions, and even frequent crashes, and we found the above strategy more robust and accurate.

2.4.4 Linear Program Size

It remains to choose reasonably small values for the problem dimensions $C \leq N$ and $K \leq n$, i.e., for the lengths of the vectors a and x, respectively that enter the LP formulation. Let K_{last} be the maximal type observed in the subsample, such that $o_k = 0$ for all $k > K_{\text{last}}$, but $o_{K_{\text{last}}} > 0$.

The probability that a type-c read generates a subtype-k read with $k > K_{\text{last}}$ is given by $p_{c,K_{\text{last}}} = \sum_{k>K_{\text{last}}} P_{c,k}^{(N,n)}$. As no such reads were observed, we need not consider values of c where $p_{c,K_{\text{last}}}$ is high, say larger than some threshold $t := 0.5$. If there were ten such type-c reads, we would then expect at least 5 with subtype larger than K_{last} instead of the observed zero reads. Thus we set

$$C := \min\left\{c \geq 1 : \sum_{k=0}^{K_{\text{last}}} P_{c,k}^{(N,n)} \leq 1 - t\right\}.$$

Given C, we might set $K := K_{\text{last}}$, but as the error terms in the objective function measure the error only up to the K-th term, we must choose a larger value to ensure that almost the whole probability mass of the hypergeometric distribution for type C is contained in the considered range for K, say up to an error of $\tau := 10^{-4}$. Thus for the given C, we set

$$K := \min\left\{\kappa \geq 1 : \sum_{k=0}^{\kappa} P_{C,k}^{(N,n)} \geq 1 - \tau\right\}.$$

As an example, consider the observation

$$o = (o_1, \ldots, o_5) = (65610, 14580, 1620, 90, 2)$$

with $n = 100000$ and $K_{\text{last}} = 5$. With $N = 10^6$, $t = 0.5$ and $\tau = 10^{-4}$, we find $C = 57$ and $K = 16$. Using more relaxed parameters $t = 0.2$ and $\tau = 10^{-3}$, we find $C = 40$ and $K = 11$.

2.4.5 Reducing Linear Program Size

The strategy described in the Section "Linear Program Size" (2.4.4) is accurate and results in small LP sizes for most real datasets. However, with extremely high duplicate rates or extreme outliers (some families with extremely many copies in the subpattern), an LP can become too large to be resolved in a few seconds. In cases (e.g., when $C \gg K > 50$) we can use the following heuristic, which sacrifices a little accuracy in favor of a much smaller problem size and higher speed.

Instead of using the real value of K_{last} of the observed occupancy vector $o = (o_1, \ldots, o_{K_{\text{last}}})$, we cut off the observed occupancy vector at a smaller, convenient index, say K^0, and perform the estimation procedure for a with this shortened vector. Accordingly, the values of n and N for the LP must be reduced (see below). At the end, we separately add back the cut-off part as follows. Since the sampling factor is n/N, the resulting subtype of most type-c families is close to $k = cn/N$. Conversely, for large enough k, a family of observed subtype k can be assumed to come from a family whose original type is close to $c = [kN/n]$, where $[x]$ is the next integer up to x. For each $k > K^0$ we increase $a_{[kN/n]}$ by o_k.

This procedure accounts for $n^0 \sum_{k>K^0} k \cdot o_k$ observed fragments and $N^0 := \sum_{k>K^0} [kN/n] \cdot o_k$ fragments in the full dataset, so that the LP must be solved with $n' = n - n^0$ and $N' = N - N^0$ instead of the original n and N.

We may expect that the error introduced by this heuristic is small because

1. the absolute number of affected families, $\sum_{k>K^0} o_k$ is small;
2. the error introduced by inferring the wrong type $c' = [kN/n]$ instead of the true c has a small effect on the resulting N' if $|c' - c|$ is small.

2.5 Software

We have implemented the point and interval estimation for the duplicate rate in a software package called `dupre` (for duplicate rate estimation). Python 3.4, along with the `PuLP` package is required. The linear programs are solved with the standard solver of `PuLP`, the Cbc (Coin-or branch and cut) command line tool (CBC-CMD) from the COIN-OR project (Computational Infrastructure for Operations Research[1]).

Our tool `dupre` is open source (MIT license) and can be obtained from Bioconda (Grüning et al., 2017) or its BitBucket code repository[2]. For

[1] http://www.coin-or.org/
[2] http://bitbucket.org/genomeinformatics/dupre

easy installation, we recommend following the README instructions in the repository. No commercial or proprietary optimization libraries are required. The software takes as input an observed occupancy vector $o = (o_1, \ldots, o_{K_{\text{last}}})$ (provided as a file or on the command line) and a target library size N and outputs point and interval estimates of the duplicate rate. On demand, other useful properties of the problem can be provided, such as the LP size (C, K) or the estimated full occupancy vector a.

The example in Section *Linear Program Size* can be run as follows (with `-v` for verbose output):

```
dupre -o 65610 14580 1620 90 2 -N 1000000 -v
```

Running `dupre --help` explains available options.

A separate tool `bam2occupancy` is available to compute the occupancy vector from reads in a BAM file. It is provided as part of the `dupre` package and additionally needs the `pysam` package to be installed.

Note that `dupre` itself is a general tool and not closely related to sequencing libraries. It is of potential use in other areas that require estimating the increase in observed diversity as sampling depth increases, such as capture-recapture experiments in biodiversity studies.

2.6 Evaluation

Three other works are dealing with the same problem. The first, `MarkDuplicates` tool from `picard-tools`, outputs, among other things, a *return-on-investment* table that estimates the useful coverage factor $f(c)$ (new original fragments) if the current coverage, including duplicates, was multiplied by c for each $c = 1, 2, \ldots, 100$. The estimation is based on Poisson statistics and not on the occupancy vector of multiple copies, which in some situations leads to inaccurate results as described in Section 2.4.1.

Second, the `preseq` tool suite of Daley and Smith, 2013 upscales the occupancy vector of the observed subsample as we do. However, our methodology differs greatly from theirs. Their estimate is based on a classical empirical estimator (Good and Toulmin, 1956), which is originally numerically unstable and has exceptionally high variance for $c \geq 2$. Daley and Smith uses rational functions to stabilize them and enable smooth extrapolation for large sequencing depths.

Third, `recon` (Kaplinsky and Arnaout, 2016) iteratively applies an EM algorithm to a growing number of Poisson-distributed clone sizes to estimate an overall parental distribution. The duplicate rate is then determined by resampling the parent distribution. It was developed to estimate the diversity of an organism's B- and T-cell repertoires. The approach has been experimentally tested for this particular application, but may not be appropriate for the general duplicate rate problem addressed by `dupre`.

TABLE 2.1: Left: average properties of different scenarios for artificial and real datasets: length of full occupancy vector a, number of original reads, total number of reads, typical duplicate rate, number of problems to solve. Right: size of evaluated real datasets. Abbreviations: M: million. For the real datasets, the values are averaged over all instances.

scenario	len(a)	$\sum_c a_c$	reads	dup.rate	#problems
d = dirac	10	1.0 M	10.0 M	90 %	2500
e = easy	7	9.7 M	10.8 M	11 %	2500
h = hard	91	1.0 M	10.0 M	90 %	2500
m = monotone	16	5.0 M	10.1 M	42 %	2500
exome	230	46.5M	55.3M	16 %	20700
rna	2662	35.8M	42.5M	16 %	11160

Also, an approach of Tauber and Haeseler, 2013 uses the Pitman Sampling Formula (Pitman, 1995) to estimate the size of the gene universe.

For the evaluation, we compare `dupre` with `preseq` and `recon` on

1. artificially generated occupancy vectors with different properties,
2. occupancy vectors obtained from several full and subsampled in-house datasets of different types: Exomes and RNA-seq.

Some of our internal records are confidential and cannot be released. However, we provide the occupancy vectors for all records in the same repository as the software and facilitate (a) the reproduction of our experiments and (b) the comparison of `dupre` estimates with those of other tools.

We describe the following four scenarios (dirac, easy, hard, monotone) with different artificial full occupancy vectors, each posing different challenges. A summary of the properties of each scenario is given in table 2.1.

For each scenario, we generate 50 instances, i.e., true occupancy vectors $a^{(s,i)}$, where s is the scenario and $i = 1, \dots, 50$ enumerates instances. For each instance, we consider five different subsampling ratios

$$\rho \in \{0.01, 0.02, 0.05, 0.10, 0.20\}.$$

For each instance and sampling ratio, we generate 10 independent subsamples and their occupancy vectors $o^{(s,i,\rho,r)}$, $r = 1, \dots, 20$ for a total of 10 000 problems.

dirac We create an occupancy vector a by setting $a_{10} := 1\,000\,000$ and $a_c := 0$ for $c \neq 10$; no randomness is involved. Thus each family has precisely ten members, for a total of ten million sequenced fragments.

easy We create an occupancy vector a whose entries drop approximately exponentially at a given rate $r < 1$. We choose a_c uniformly around a target number T_c, between $T_c/2$ and $2\,T_c$, and then set $T_{c+1} = r \cdot T_c$. We here use

$T_1 := 7\,000\,000$ and $r := 0.1$. This creates short occupancy vectors with steeply decreasing values.

hard This scenario is similar to the easy one, but we use $T_1 := 80\,000$ and $r := 0.9$. We obtain long vectors whose entries are of similar size to each other.

monotone We set $a_1 = 2\,500\,000$ and then each a_c randomly uniformly distributed between 0 and a_{c-1} until we reach zero. This creates monotone decreasing occupancy vector a.

We tested both tools additionally on real occupancy vectors from BAM files of projects at the University Hospital Essen. The available data sets consisted of 414 exome sequencing and 232 RNA-Seq samples. The properties (library size, length of a, duplicate rate) varied considerably within each scenario and led to a variety of problems of different difficulty and size. We computed the true duplicate rates from the given BAM files with mapped reads and then created 10 random subsamples for each of the five sampling rates $\rho \in \{0.01, 0.02, 0.05, 0.10, 0.20\}$ for a total of $(414 + 232 + 92) \cdot 10 \cdot 5 = 32\,300$ problems to solve.

Results

Each observed occupancy vector $o^{(s,i,\rho,r)}$, together with the known full sample size N of $a^{(s,i)}$ serves as input to `dupre`, `preseq` and `recon`. For each problem, each tool outputs a point estimate and an interval estimate for the duplicate rate; `dupre` and `preseq` claim to provide approximate 95% confidence intervals. We evaluate according to the following criteria:

1. number of instances solved without errors or crashes
2. the difference between the point estimate of the duplicate rate and the known true duplicate rate, which should be close to zero and of small variation
3. the number of times that the true duplicate rate is in the estimated confidence interval, which should be close to 95%

Since `recon` reports a confidence interval for the estimated parent distribution, but not for its downsampling, the confidence interval precision is not given for `recon`).

Evaluations were automated with Snakemake (Köster and Rahmann, 2012) and took a few days overall, including instance creation. Every single problem is solved in a few seconds up to a minute with `dupre`. The running time is longer for smaller ρ because the LP size increases with $1/\rho^2$. As shown below, `preseq` had convergence problems on several instances and took extremely long times for some of them. In practice, running time for a single instance is not an issue for none of the three tools.

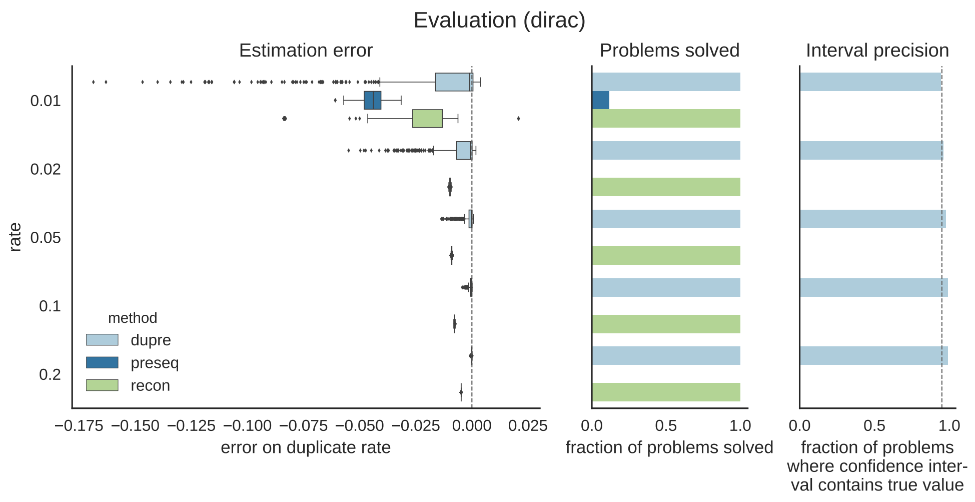


FIGURE 2.5: Results for scenario *dirac*. Left: Error of estimated duplicate rate (boxplots) for `dupre` (light blue), `preseq` (dark blue) and `recon` (green) with median (centered bars), interquartile range (boxes) and minimum and maximum (black bars) except outliers (blue crosses). An ideal estimate has an error of zero (dashed vertical line), while positive values state that the tool overestimates the true duplicate rate. Center: Fraction of problems solved for each subsampling rate $\rho \in \{0.01, 0.02, 0.05, 0.10, 0.20\}$ and each tool. Right: Fraction of solved instances where the output confidence interval contains the true solution. As `dupre` and `preseq` methods claim to output 95% confidence intervals, this should be approximately 0.95 (dashed vertical line). `preseq` produced no solutions for $\rho \geq 0.02$ and few for $\rho = 0.01$. Due to the missing estimation this value is omitted for `recon`.

Figure 2.5 shows the results for the *dirac* record. `Dupre` resolves each instance and keeps the 95% confidence interval promise for each partial subsample rate. The point estimation errors of `dupre` are nearly zero, except for some outliers at the lowest partial sample rates, which we expected. Unfortunately, `preseq` is not able to solve most of these problems, either because it rejects the small input vector or returns an error during the calculation. While `preseq` solves only a few problems with $\rho = 0.01$ where it underestimates the exact duplicate rate by about 0.05, `recon` can solve all problems with good performance. It slightly underestimates the duplication rate across the entire spectrum of downsampling rates.

Figure 2.6 (a) shows the results for the data set *easy*. Also here `preseq` cannot solve most problems and does not keep the 95% confidence interval promise. For solved problems, however, the estimate is very accurate. In fact, both tools estimate the duplicate rate accurately, even for small ρ. The larger variation in the error in `dupre` is partly due to the much larger number of resolved instances. On the other hand, `recon` can solve all problems but overestimates the duplication rate for small $\rho < 0.1$ strongly.

On the record "hard" (Figure 2.6) (b)), `preseq` and `recon` improve their results: `Preseq` solves most but not all of them and both competitors have very small estimation errors. The confidence intervals of preseq are somewhat too narrow (95% is not reached). The estimation errors of `dupre` show a larger deviation around zero, especially with low ρ, where also confidence intervals are a bit too wide.

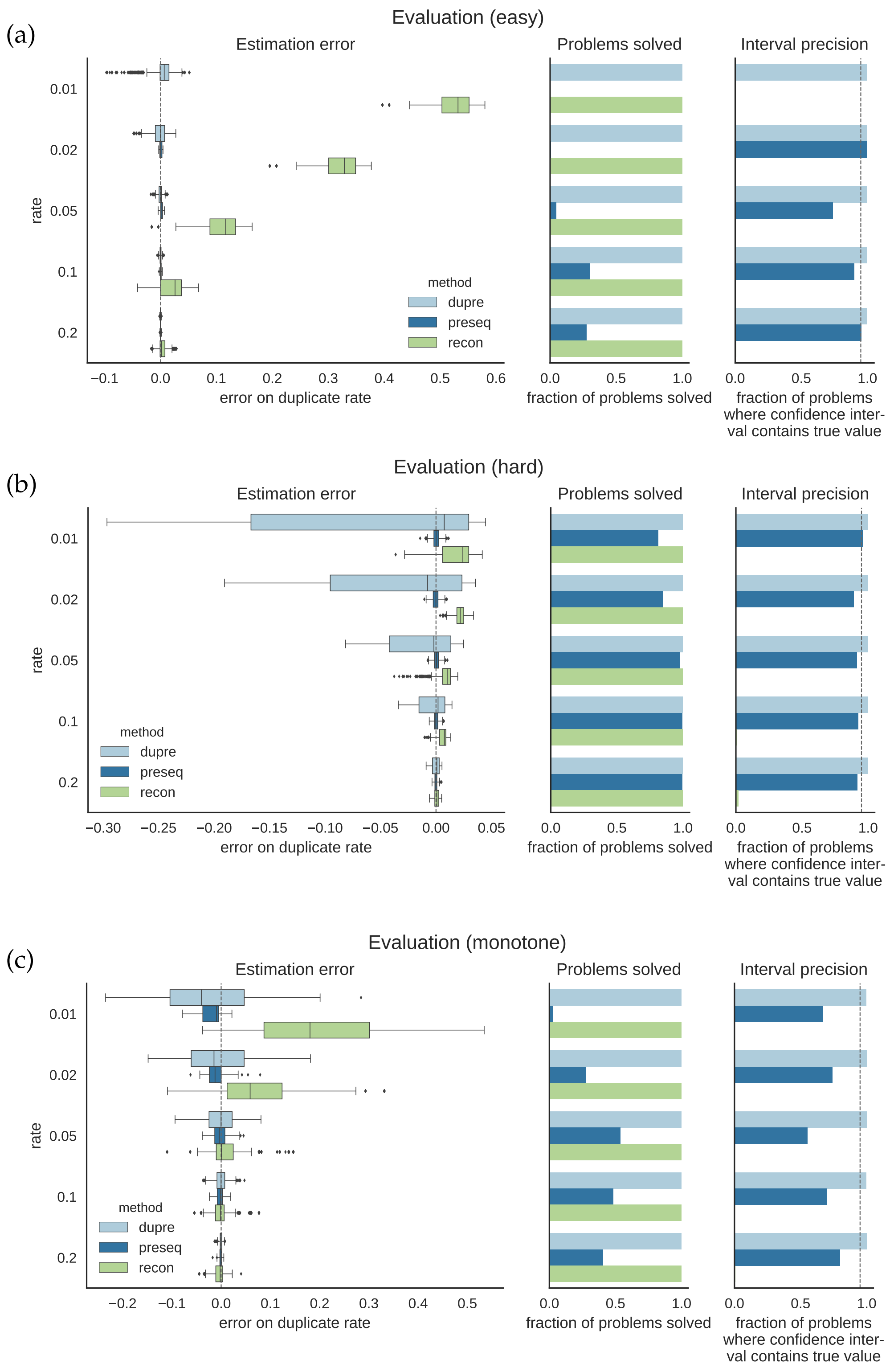


FIGURE 2.6: Results for scenario (a) *easy*, (b) *hard* and (c) *monotone*; cf. Figure 2.5.

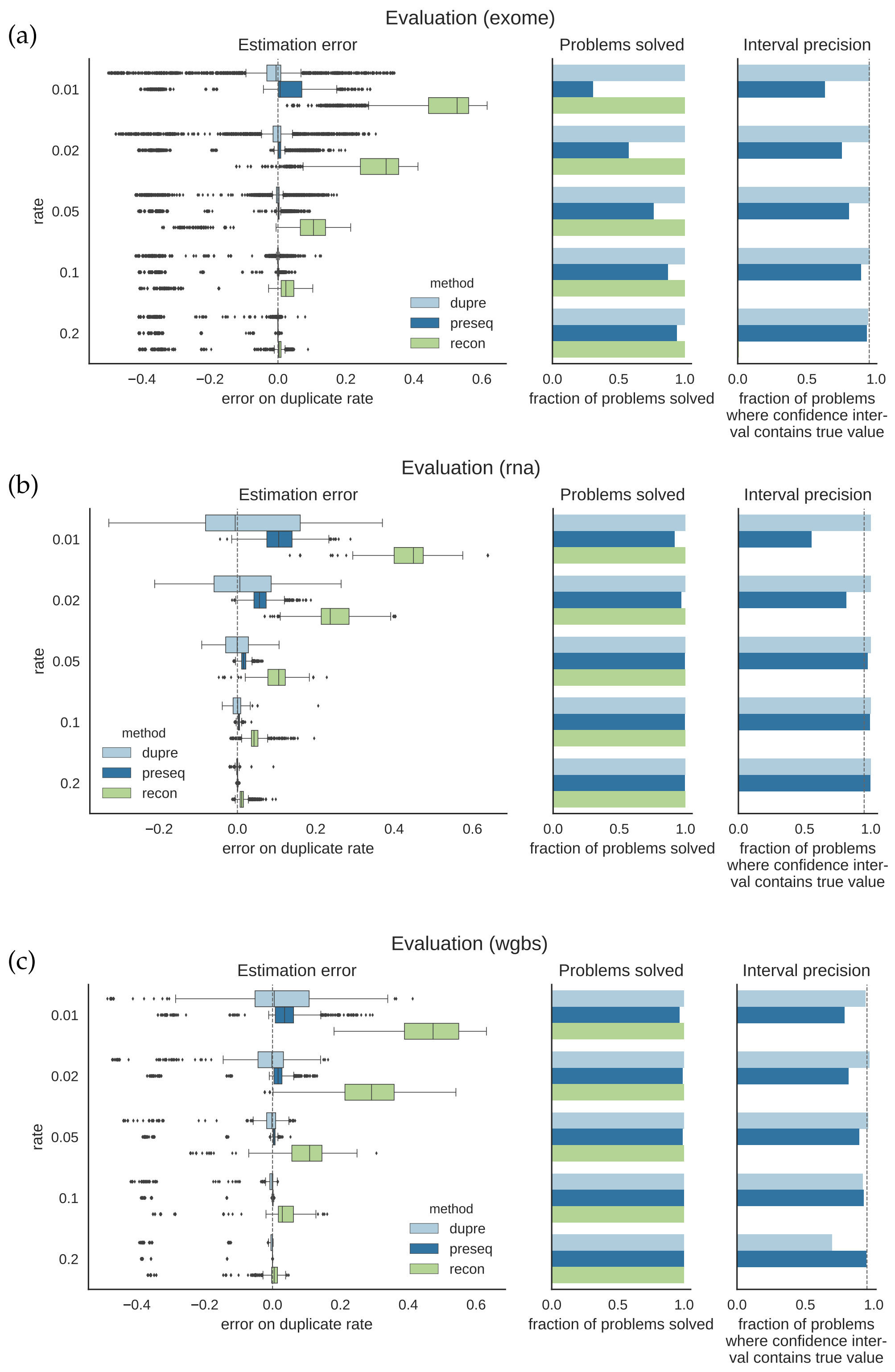


FIGURE 2.7: Results for (a) real exome, (b) RNA-seq, (c) WGBS datasets; cf. Figure 2.5.

For the record "Monoton" (Figure 2.6) (c)) we get a similar picture for `dupre` and `preseq`. `Dupre` solves all instances, gives good (slightly too wide) confidence intervals and shows a larger variation of errors, while `preseq` does not solve most instances and gives narrow confidence intervals, but achieves excellent point estimates. `Recon` overestimates again for small $\rho < 0.1$.

The results are also reflected in the first real *exome* dataset. Again, `preseq` only solves a few problems for small ρ and gives too small confidence bounds. Both `preseq` and `dupre` can estimate the duplicate rate very accurately. The middle and lower and upper quartiles of the error are very close to zero, but we find that both tools produce multiple outliers, even for large ρ.

In the RNA Seq record (*rna*), `preseq` solves almost all problems, but still doesn't give exact confidence intervals for small ρ. The estimates of `dupre` have a small distortion (medians are on the zero line), but a large variation for $\rho < 0.05$, while the estimates of `preseq` are too high with a smaller variation. `recon` overestimates again for small rho. However, for $\rho \geq 0.1$ all three tools perform well.

The evaluation on the WGBS dataset shows similar results as in the case of the RNA-Seq dataset. However, `dupre` does not give sufficiently large confidence intervals for $\rho = 0.2$. This is the only case where `dupre` underestimates the interval accuracy. Interestingly, `recon` shows almost exactly the same differences as before for *rna* and *exome* of about +0.4 for a rate of 0.01, about +0.25 for a rate of 0.02, +0.1 for a rate of 0.05 and +0.05 for a rate of 0.1. This is probably caused by similar distribution forms in all three sets of real data and may allow the application of a correction factor depending on the downsampling rate rho.

2.7 Summary

In this chapter, we present an optimization approach to estimate the duplicate rate in a large sequencing library from a small subsample. Our approach uses a linear program based on hypergeometric probabilities that capture the subsampling dynamics to estimate the occupancy vector of the full sample from that of the subsample. Our method is useful in a variety of situations: it provides reliable estimates and confidence intervals, while the existing `presseq` method does not solve multiple cases and often generates too narrow confidence intervals. The second method of `recon`, developed initially to estimate cell repertoires, overestimates the duplication rate for small subsampling rates in the vast majority of cases. However, we found that the error is very constant depending on the rate for real read data. Correcting this error with an appropriate function is possible.

While our primary motivation for developing this approach was to explore the benefits of increasing sequencing depth for a particular library, it solves a general problem with biodiversity applications, as evidenced by the traditional statistical research on this problem (Good and Toulmin, 1956).

Chapter 3

Methylome Processing

Adaptations to existing NGS methods allow the sequencing of different parts or properties of the genetic material (see Section 1.3), e.g., methylation. The current state of the art for determining DNA methylation levels at the whole genome level is whole genome bisulfite sequencing (WGBS). Treatment of DNA with sodium bisulfite (HSO_3^-) converts unmethylated cytosine into uracil, which is recognized and output as thymine by the sequencer. As a result, bisulfite sequencing and its subsequent *in silico* steps produce aligned paired reads, most unmethylated cytosines being substituted by thymine - a so-called *methylome*. The mean degree of methylation for each CpG can then be calculated from the number of thymines and cytosines measured in the reads overlapping at this point. This leads to some computational challenges.

Due to the substituted cytosines, the reads may show several mismatches compared to a specific reference genome. The aligner used must be modified to handle such reads.

For methylomes, the information of interest is the ratio of methylated Cs in the CpG context to its coverage, which is typically extracted from an alignment by a methylation caller. The most popular caller `Bis-SNP` (Liu et al., 2012) uses a method to extract bisulfite substitutions from a list of genetic variants generated by `The Genome Analysis Toolkit` (`GATK`, McKenna et al., 2010). Allelic frequencies of the identified cytosines in the CpG context then lead directly to their degree of methylation. However, the whole genome variant calling on bisulfite reads with high coverage is very computationally intensive. The data can contain several hundred billion bases, and since each position is covered, the program creates 3 billion pileups in a first step. Also, the substitution of almost all cytosines present in the genome led to a massive list of variants and probability calculations for each of these variants. Even for a single data set and parallel execution, the calculation can take days on suitable server hardware.

This chapter describes general concepts for the alignment of bisulfite-modified reads and our newly developed tool `Calling and Analysing Methylation Levels.` (`camel`). This tool reduces the time required to determine the methylation level by creating an index in a previous step and then considering only predefined positions for these values.

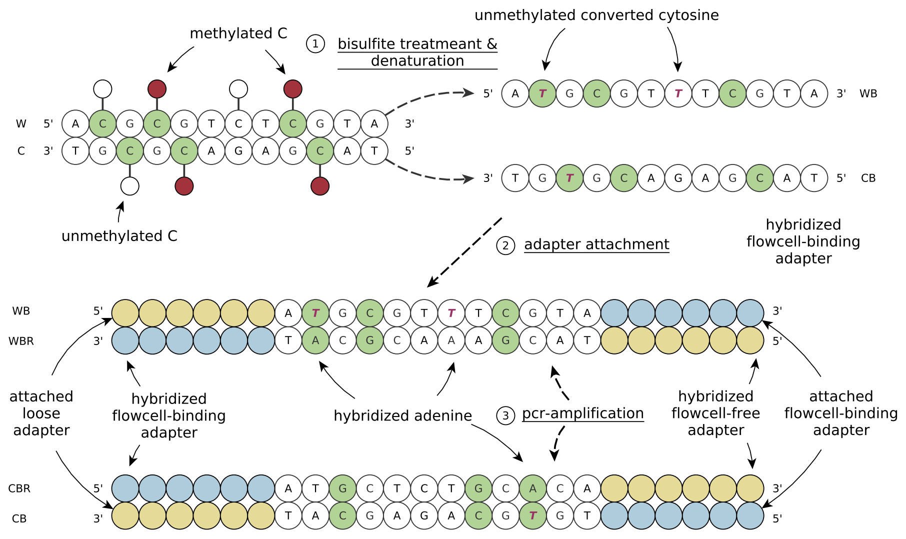

methylated C
① bisulfite treatmeant & denaturation
unmethylated converted cytosine
unmethylated C
② adapter attachment
hybridized flowcell-binding adapter
attached loose adapter
hybridized flowcell-binding adapter
hybridized adenine
③ pcr-amplification
hybridized flowcell-free adapter
attached flowcell-binding adapter
W
C
WB
CB
WBR
CBR

FIGURE 3.1: The bisulfite sequencing process. ① A double-stranded DNA, consisting of the Watson (W) and its reverse-complementary Crick (C) strand is denaturated and treated with bisulfite. In this process, unmethylated cytosine (with an attached white circle) is converted to uracil while methylated cytosine (with an attached red circle) is unaffected. Because the sequencer does not distinguish between uracil and thymine, it notes Uracil as T. The resulting strands are named WB (Watson bisulfite treated) and CB (Crick bisulfite treated). ② The 3′ ends of WB and CB are extended by an adapter sequence (blue), which binds to the flowcell. The 5′ ends are extended by a second (free) adapter. ③ As described in Section 1.3, the reverse complement is hybridized to WB and CB by PCR-amplification, resulting in two fragments named WBR and CBR, which binds to the second type of flowcell probes.

3.1 Bisulfite Read Mapping

The methylome sequencing process is similar to the unmodified DNA sequencing described in Section 1.3 except for an additional DNA bisulfite treatment step. This leads to a DNA change, which leads to alignment problems. In the following, we describe these problems and their solution.

3.1.1 Bisulfite Treatment

DNA consists of two reverse complementary strands: the *Watson* (*W*) and the *Crick*. (C) Strand, named after James Watson and Francis Crick, two of the discoverers of DNA structure. A bisulfite treatment causes non-methylated cytosines on both strands to transform independently into uracil. Note that this eliminates the property of reverse complementarity. Sequencing adapters are added to the transformed strands to form the fragments, often referred to as *WB* and *CB*, which then bind to the flow cell probes and are amplified by PCR amplification. The reverse complementary fragments that hybridize to *WB* and *CB* are called *WBR* and *CB*, respectively. *CBR*. Now hybridization bases in *WBR* and *CBR* are adenine instead of guanine at positions where unmethylated cytosine in *WB* and *CB* has been substituted with thymine. After the last PCR cycle, *WBR* and

CBR are washed away (identified by the respective adapters). Therefore, the first reads of the pairs are sequenced by synthesis only on the *WB* and *CB* fragments, the second reads on their reverse complements (*WBR* and *CBR*). Figure 3.1 illustrates the process of bisulfite sequencing.

Bisulfite sequencing generates reads from *four* different sequences instead of the *two* different strands common in traditional sequencing. Depending on the methylation state and strand, cytosine may be substituted by thymine and guanine by adenine instead of the two different strands of WGS. This leads to read aligning problems. SNPs, indels or sequencing errors cause mismatches compared to the reference. The read mapper punishes these rare events in its internal alignment score. Compared to genetic mismatches or mismatches caused by sequencing errors, cytosine/thymine substitutions are quite common, and a single read usually contains several such modifications, resulting in a high number of mismatches. In general, fault-tolerant read mappers can handle multiple single-base substitutions, but any deviation reduces the calculated mapping quality. This leads to distortions in the direction of those reads that contain methylated cytosines, with the bases matching the reference after bisulfite treatment. Besides, the alignment is discarded if the score falls below a certain threshold. This, in turn, causes the mapper to generate lousy quality alignments or unmappable reads. For unbiased alignments, an Aligner must correctly handle methylation-induced C $\rightarrow$ T and G $\rightarrow$ A misalignments by ignoring them.

There are two ways not to count these substitutions as mismatch: wildcard and three-letter alignments. We illustrate these two methods with an example from Bock (2012). Figure 3.2 (above) shows the example setup with eight reads generated by WBGS that were aligned by both methods. For simplicity, the example deals with single-end forward reads only.

3.1.2 Wildcard Alignment

Unlike ordinary mappers, wildcard aligners can map both nucleotides – C and T – directly to a specific reference base without affecting the mapping score. They often support the complete IUPAC code (see Section 1.2). In the case of bisulfite mapping, the codes Y = {C, T} and R = {A, G} are important. Before indexing, the specified reference sequence is changed by replacing each C with Y, so that read accesses with converted and unconverted Cs can be aligned at this position. Instead of the IUPAC code, a placeholder alignment can also be realized by completely ignoring all Cs in the reference. This is implemented either by specially developed algorithms that mask these bases during their reference indexing or mapping step or by copying the reference and replacing C with N. Figure 3.2 (mid) shows the wildcard alignment approach for the example presented. Here, the wildcard approach *multiple* recognizes optimal mapping positions for one of our eight example reads, resulting in distorted methylation values. Popular wildcard aligners for bisulfite treated DNA are, for example, `BSMAP`. (Xi and Li, 2009), `RMAPBS` (Smith et al., 2009), `GSNAP` (Wu and Nacu, 2010) or `RRRBSMAP`. (Xi et al., 2012).

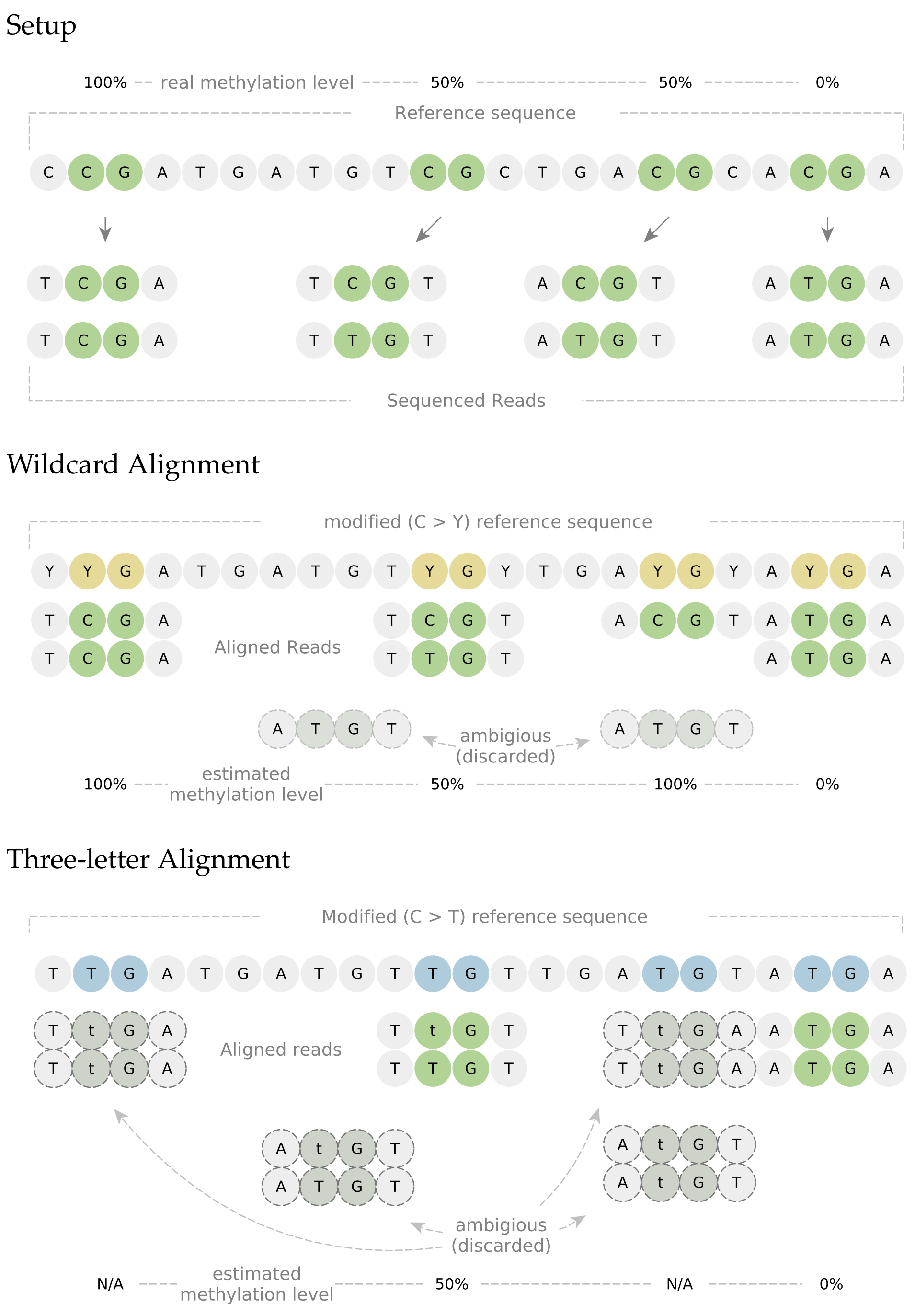

Setup
100% real methylation level 50% 50% 0%
Reference sequence
C C G A T G A T G T C G C T G A C G C A C G A
T C G A
T C G A
T C G T
T T G T
A C G T
A T G T
A T G A
A T G A
Sequenced Reads
Wildcard Alignment
modified (C > Y) reference sequence
Y Y G A T G A T G T Y G Y T G A Y G Y A Y G A
Aligned Reads
ambigious (discarded)
100% estimated methylation level 50% 100% 0%
Three-letter Alignment
Modified (C > T) reference sequence
T T G A T G A T G T T G T T G A T G T A T G A
Aligned reads
ambigious (discarded)
N/A estimated methylation level 50% N/A 0%

FIGURE 3.2: Example comparison of wildcard and three-letter alignment. Top: Reference sequence and eight reads consisting of four bases generated by WGBS and assumed (real) methylation levels. C → T substitutions are applied on unmethylated cytosines. Center: Wildcard alignment of the example reads. All Cs of the references sequence are replaced by Y, allowing the wildcard aligner to map reads with C or T at these specific positions. Each four-base snippet represents one read alignment. One read maps to multiple positions (ambiguous) and is therefore ignored from methylation level estimation. Additional estimations based on mapped reads of methylation levels are shown. Bottom: Three-letter Alignment: All Cs of the references and read sequences are replaced by T. Four reads map to multiple positions (ambiguous) and are ignored.

3.1.3 Three-letter alignment

A second approach uses two artificial references. The first reference R_f is obtained by C $\rightarrow$ T substitutions representing *WB* string sequences, and the second reference R_r by G $\rightarrow$ A substitutions corresponding to *CBR*. Since the alphabet in principle reduced from four to three letters per string, this approach is called *three-letter mapping*. R_f and R_r can then be used with a general read mapper.

As the name implies, the paired-end sequencing creates pairs of reads for each fragment. The wash step before the first sequencing process ensures that the origin of the first read p_1 of each pair is $P = (p_1, p_2)$ either *WB* or *CB*. In both cases, only C-T substitution may have occurred. Due to convolution, amplification (by backward complementation) and the additional washing step, the origin of every second p_2 read is either *WBR* or *CBR*, with possible G to A substitutions.

The general approach is to replace each C by T in p_1 and each G by A in p_2 to obtain two new sequences p'_1 and p'_2. $P' = (p'_1, p'_2)$ is then aligned by a regular read aligner to the artificial references R_f and R_r. There are two ways for the mapper to align the read results:

1. p'_1 maps directly to R_f and the reverse complement of p'_2 to R_f
2. the reverse complement of p'_1 maps to R_r and p'_2 directly to R_r

The final output is the combination of (original) sequences of P and the alignment information of P'.

Depending on the maximum reference size of the mapper, concatenating R_f and R_r to a unified reference R_u of about 6 billion in length can allow alignment to both references in a single step. An aligner index is typically able to address $2^{32} \approx 4.3$ billion positions, more than the 3 billion base pairs of the human genome, but not enough for a unified reference. If the mapper supports 64-bit indexing, R_u can be indexed directly and used as a single combined reference. If only 32-bit indexing is supported, the mapping for R_f and R_r must be performed independently of each other by two processes, each receiving a copy of the transformed reads. Both assign Phred-scaled quality values to their alignment of the same reads pair, representing the probability that the read operation was mapped at the wrong position. Note that mappers are often able to generate alignments (of low quality and complex sequence editing operations) even with an unsuitable reference. The final alignment is calculated by comparing the results of both independent mapping processes. For each read pair, the quality values are calculated, and the alignment with the higher value is taken as the final result. Figure 3.3 illustrates the workflow for mappers with and without 64-bit support. Popular tools of this category are BS Seeker (Chen et al., 2010), Bismark (Krueger and Andrews, 2011), MethylCoder (Pedersen et al., 2011), BatMeth (Lim et al., 2012), BRAT-BW (Harris et al., 2012) or BWA-Meth (Pedersen et al., 2014).

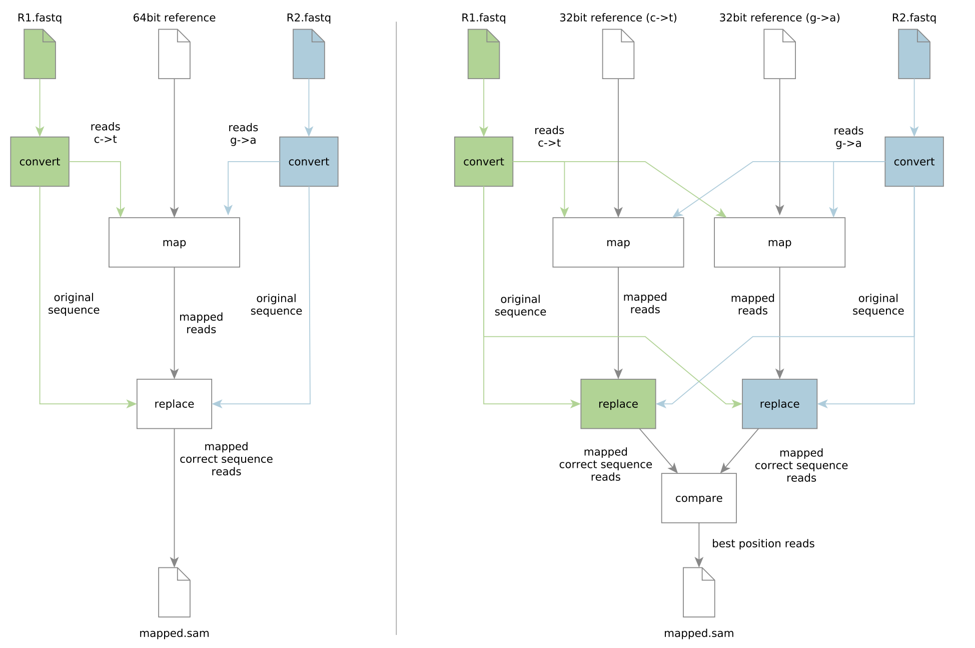


FIGURE 3.3: The basic structure of a three-letter alignment. Left: the input is a 64-bit reference. The fastq files get transformed by a C to T, and a G to A transformation, mapped by the mapper and the sequences are reconverted to its originals. Right: the input are two 32 bit references. Two independent mapping processes map the duplicated transformed reads to each of the references. The processed re-transformed reads are pairwise compared and chosen by the highest quality.

The error rates in our example for both approaches (wildcard and three-letter approach) are quite high, with 50% false methylation values for the wildcard and 50% of reads lost in the three-letter alignment. In real data, the effect of ambiguous alignments is much smaller than in this example. The more extended sequences and link information usually lead to unambiguous mapping positions, even with three-letter alignments. Even if a small bias is present, small distortions of the methylation values for certain CpGs usually do not influence the results in the downstream analysis if the degree of methylation of all samples is estimated equally.

Three letter alignment is a general principle and can be used in conjunction with almost any read mapper. This makes it possible to build a universal methylation wrapper that does not rely on a single specific aligner. We have started to develop such a tool, but it shows that existing implementations of three-letter aligners often use the specific functions of the aligner to improve system performance. This would be, for example, storing the original sequence recovery information as a suffix of the read name or using interleaved paired-end reads. Since mapping speed is one of the most important metrics, a universal wrapper cannot compete with other tools, and we rejected this idea. We recommend the use of one of the existing and published solutions based on established standard mappers. However, the described principle can serve as a blueprint to create wrappers for each selected mapper.

3.2 Calling methylation values

Methylome sequencing is used to determine the binary information on the methylation state of each methylated or unmethylated CpG. Today's sequencing is typically based on millions of cells, each with its unique methylation pattern. Due to shotgun sequencing and the overlapping of Reads n_c, binary methylation states are measured for each CpG c with coverage n_c. The degree of methylation for c is an average measurement of the total binary states and can be expressed in various ways.

Let m_c be the count of methylated and u_c the count of unmethylated states at CpG c, with $m_c + u_c = n_c$. So called beta values (β-values)

$$\beta_c = \frac{m_c}{m_c + u_c} = \frac{m_c}{n_c}$$

describe the ratio of methylated reads compared to the coverage of a CpG. A second option, often used for methylation arrays, are M-values

$$M_c = \log_2\left(\frac{m_c + \alpha}{u_c + \alpha}\right)$$

with additional pseudo count α to prevent drastic changes for small measurements. While β-values allow for an intuitive interpretation of methylation (*% methylation for a given site*), the M-value may provide some insight into the distribution of methylation across the genome that is difficult to visualize with the β-value (Du et al., 2010). Both values can be transformed into one another (assuming $\alpha = 0$) by

$$\beta_c = \frac{2^{M_c}}{2^{M_c} + 1}; \quad M_c = \log_2\left(\frac{\beta_c}{1 - \beta_c}\right)$$

AtAt this point, we mention M-values only for the sake of completeness. It is common to base the sequencing of whole genome bisulfite data on β values. Also in this thesis, the term methylation refers to β values, unless otherwise stated. Regardless of whether the values are β or M, the counts m_c and u_c for $c \in C$ must be measured using a given read orientation. A so-called methylation caller performs this task.

WGBS generates an enormous amount of information consisting of more than a billion mapped reads and a hundred billion bases covering most genomic positions. One method to estimate these counts is to use a variant caller that recognizes genetic variants based on the reads provided and the reference genome. Since unmethylated cytosine has been converted into thymine, such position is recognized by such a program as SNP. A second step then identifies Cs in a CpG context based on the reference and SNP lists and uses the SNP information to estimate m_c and u_c. For example, this approach is applied by `Bis-Snp` (Liu et al., 2012). The main disadvantage of this method is the costly necessary step to create a pileup for each genomic position. These form the basis of SNP detection. The identification of genetic variants on the whole genome as pre-processing is computationally intensive and can take several days depending on available resources

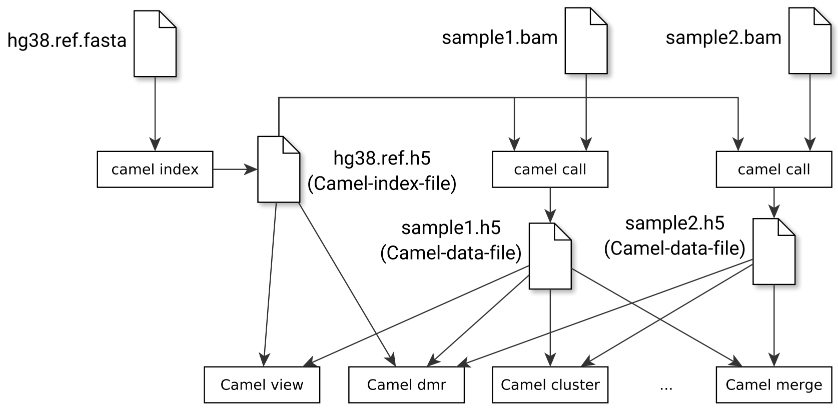


FIGURE 3.4: Basic concept of camel analysis. Camel stores all required data in a static index-file and dynamic data-file for each sample. Most of the provided downstream analysis required these files. In this example *camel index* uses the reference (*hg38.ref.fasta*) ro create the camel-index-file (*hg38.ref.h5*), which contains positions for potential methylation. The index is then used by *camel call* in combination with two alignments (*sample1.bam* and *sample2.bam*) to create two camel-data-files (*sample1.h5* and *sample2.h5*). These files then contain the methylation values associated with the given positions. See Figure 3.5 for details about the camel-data- and Camel-index-files.

and sample coverage.

We present `camel` based on fixed CpG positions instead of calling individual CpGs in each sample as executed by `bis-Snp` and other tools. This approach drastically shortens the runtime compared to the above method and has some additional computational advantages. `Camel` can handle not only WGBS but also NOMe-seq data where the degree of methylation of cytosine is measured in a GpC context (see Chapter 1.3 for details). Its general concept relies on two separate files: an index file – the *camel-index* – with all CpGs and GpC positions and a file with the data for each sample – the *camel-data-file*. In the following we describe details of this structure, the `camel` call process, additional features and their advantages.

3.2.1 The Camel Index

Once calculated from the Fasta reference, the *camel index* holds all cytosine positions with CpG context C_h and GpC context G_h for the chromosome h. The analysis of NOMe-seq methylomes requires such GpC positions. `Camel` supports two different modes: a normal mode, in which counts are processed in the CpG context, and a NOMe mode, in which the methylation of cytosine is simultaneously handled in the CpG and GpC context. In the following, we describe `camel` in NOMe mode, unless otherwise stated.

For both the index and the following measurements, hierarchically structured hdf5 files (see Section 1.7) are used. The positions in the index are stored as a group for each h chromosome. Each group contains a $|C_h|$ dataset with the CpG positions of the reference and a $|G_h|$ dataset for GpC positions.

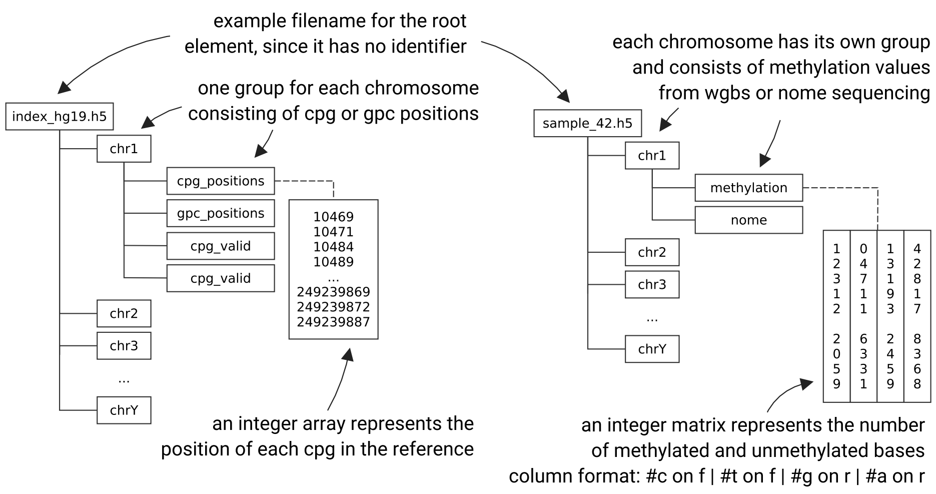

example filename for the root element, since it has no identifier
one group for each chromosome consisting of cpg or gpc positions
index_hg19.h5
chr1
cpg_positions
gpc_positions
cpg_valid
cpg_valid
chr2
chr3
...
chrY
10469
10471
10484
10489
...
249239869
249239872
249239887
an integer array represents the position of each cpg in the reference
each chromosome has its own group and consists of methylation values from wgbs or nome sequencing
sample_42.h5
chr1
methylation
nome
chr2
chr3
...
chrY
an integer matrix represents the number of methylated and unmethylated bases
column format: #c on f | #t on f | #g on r | #a on r

FIGURE 3.5: Hdf5 structure of the camel index (left) and the camel data file (right).For each chromosome, the index is divided into groups containing the CpG and GpC positions (for NOMe-seq) of the reference genome.There are two additional arrays with information about overlaps between GpC and CpGs (*cpg_valid* and *gpc_valid*). The origin of the methylation information in such an ambiguous context (see also Figure 3.6) cannot be determined and must, therefore, be ignored in most downstream analyses. Position corresponding CpG and GpC methylation values are stored as four-tuples (methylated and unmethylated values on forward and backward strands) in the camel data file under *methylation* and *nome*.

In the case of NOMe-seq, methylation of cytosine in the context of GpCpGs is caused by either natural CpG methyltransferase or artificial GpC methyltransferase. Figure 3.6 shows an example sequence where this problem occurs. To exclude such an ambiguous cytosine from downstream analyses, we have added two additional Boolean datasets of $|C_h| \times 2$ and $|G_h| \times 2$, which store whether a corresponding cytosine position on the forward or backward (or both) position is ambiguous. These matrices are used in downstream analyses to mask and ignore such positions. The left side of Figure 3.4 shows the index hdf5 structure.

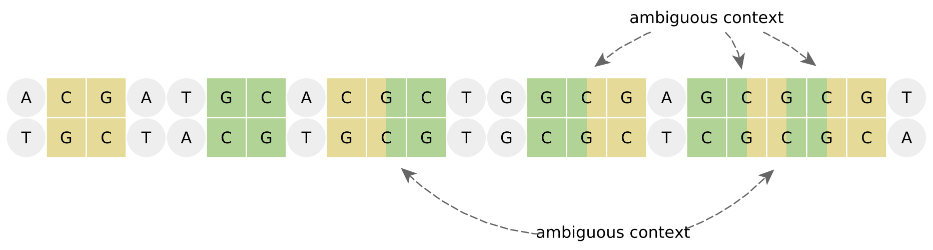

ambiguous context
A C G A T G C A C G C T G G C G A G C G C G T
T G C T A C G T G C G T G C G C T C G C G C A
ambiguous context

FIGURE 3.6: Example for ambiguous contexts of cytosines with CpG context (yellow) and GpC context (green). Some bases (dual color: yellow and green) belong to both contexts. Cytosines in both contexts are called ambiguous. Their methylation origin is indistinguishable when NOMe is applied.

3.2.2 The Camel-data-file

While the index holds all CpG positions, a corresponding *camel-data-file* contains counts m_h and u_h for one specific sample. Similar to the index structure, each chromosome h gets its own group, containing a matrix $|C_h| \times 4$ holding the counts of ...

- ... methylated C on the forward strand (m_h^f)
- ... unmethylated C on the forward strand (u_h^f)
- ... methylated C on the reverse strand (m_h^r)
- ... unmethylated C on the reverse strand (u_h^r)

Row i, with $0 \leq i < |C_h|$, addresses the counts for the i-th CpG on chromosome h. The required values to calculate the methylation level are given by $m_h = m_h^f + m_h^r$ and $u_h = u_h^f + u_h^r$. When created in NOMe mode it contains a second matrix of the same structure and counts for methylation in GpC context. Figure 3.5 shows the hdf5 structure of a camel-data-file. The similar structure of index and data-file allows unifying both into a single file storing data and index information and still keeping a simple structure.

3.2.3 The Calling Process

`Camel`'s calling function is responsible for generating a camel-data-file from a given methylation alignment and the previously calculated camel-index. Instead of building the pileup of covering bases for each of the 3 million base pairs, `camel` determines the necessary counts read-wise. Two read properties are of importance. For a read r, let

$$first(r) = \begin{cases} 1, & \text{if read is the first sequenced read in a pair.} \\ 0, & \text{otherwise.} \end{cases} \tag{3.1}$$

and

$$rev(r) = \begin{cases} 1, & \text{if the read is reverse complementary mapped.} \\ 0, & \text{otherwise.} \end{cases} \tag{3.2}$$

Both information can be seen from the alignment. The first step of the algorithm identifies the strand origin and which nucleotide was substituted by which. Four different combinations are possible.

	$first$	$\neg first$
rev	G → A / R_r	C → T / R_f
$\neg rev$	C → T / R_f	G → A / R_r

- $first(r) \wedge \neg rev(r)$. The first read always substitutes C by T, and therefore a non-reverse complementary alignment of the first read always refers to R_f as a reference with a C to T substitution.

- $first(r) \wedge rev(r)$. If the first read maps reverse alignment the used reference is R_r and due to the reverse complemented sequence, the substitution is G to A.
- $\neg first(r) \wedge \neg rev(r)$. A non-reverse mapping of a second read aligns to R_r, due to the additional folding step before sequencing and therefore a G to A substitution happened.
- $\neg first(r) \wedge rev(r)$. The reverse mapping of a second read aligns to R_f with a C to T substitution.

Algorithm 1 shows a simple function to identify substitution bases and mapped strand.

```
Data: a read
Result: origin base, substitution base, mapped strand (0 for $r$, 1 for $f$)
Function orientation(Read r)
    if first(r) == rev(r) then
        return G, A, 1
    else
        return C, T, 0
```

ALGORITHM 1: Determine the subtitution bases and the mapped strand

```
Data: a read, a position
Result: a sequence position relative to the read start with respect to its
        edit operations, given an absolute position or -1 if the position
        is inside a deletion or -2 if the position is outside the sequence
        boundaries
Function relative_position(Read r, Integer position)
    relative_pos = position - read.position
    curr_pos = 0
    foreach type, n in cigar do
        if curr_pos + n $\leq$ relative_pos then
            if type == "deletion" or type == "soft-clipping" then
                relative_pos -= n // deleted bases
            else if type == "insertion" then
                relative_pos += n // inserted bases
            else
                curr_pos += n // matching bases
        else
            if type == "deletion" then
                return Null // cpg inside deletion
            else
                return
            relative_pos
    return -2
```

ALGORITHM 2: Determine the subtitution bases and the mapped strand

The procedure itself searches for read overlapping CpGs using the indices and position as well as edit operations of the read. Insertions, deletions and splitting of the read must be taken into account, since CpGs that are actually covered could possibly be omitted. The code in algorithm 2 returns the relative position of the read sequence base at an absolute position.

The functions of algorithm 1 and 2 extract the bases covered by CpGs and compare them with the expected base (C/T or G/A) to identify the binary methylation state of each read sequence at that position.

Starting with a binary search, the main loop of the algorithm 3 increases the CpG positions and searches for bases that overlap the respective CpG. Strand-specific methylation counts for the given position are then incremented based on the observed bases. The first four columns of the resulting $C_h \times 6$ matrix for each h chromosome form the camel-data-file as described in Section 3.2.2. The last two additional columns s_h^f and s_h^r represent non-methylation related base observations. Positions with high percentages (default > 0.2 in one of the strands) are likely to be affected by SNPs and removed by zeroing the respective counts.

```
Function Main(Reads reads, CamelIndex index)
    foreach r ∈ reads do
        ref, alt, strand = orientation(r)
        // note:  strand ∈ {0,1}
        indices = index.cpg_positions[r.chromosome]
        p = binary_search_index(r.position, indices)
        while rel_pos = relative_position(indices[p] + strand) > -2 do
            p++
            if rel_pos == -1 then
                // inside deletion, ignore CpG
                continue
            if r.sequence[rel_pos] == ref then
                // strand specific methylated CpG
                // index depending on strand ∈ {0,2}
                count[r.chrom][0 + strand · 2] += 1
            else if r.sequence[rel_pos] == alt then
                // strand specific unmethylated CpG
                // index depending on strand ∈ {1,3}
                count[r.chrom][1 + strand · 2] += 1
            else
                // strand specific other base (SNP)
                // index depending on strand ∈ {4,5}
                count[r.chrom][4 + strand] += 1
```

ALGORITHM 3: Main loop of the algorithm, iterating over all (unsorted) reads and comparing observed to expected bases to store methylation counts and SNP counts.

Fixed CpG-positions also allow extracting methylation levels of unsorted and unindexed alignments, as illustrated in Figure 3.7. Due to otherwise

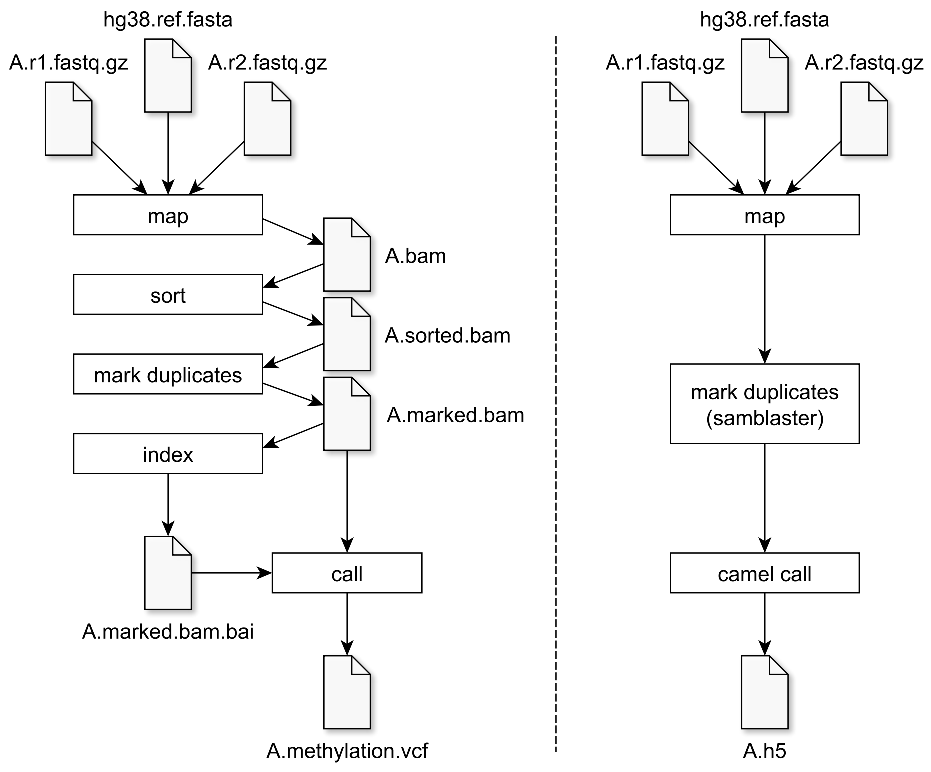


FIGURE 3.7: The call process of `Camel` (right) compared to a call process with a conventional tool (left). The ability to use unsorted reads and duplicate detection allows direct extraction of methylation levels from a streamed alignment.

missing PCR-duplication information, we added a preprocessing step that mark duplicates by `samblaster` (Faust and Hall, 2014), which can mark them without the requirement of position sorted reads.

3.3 DMR calling

Scientific research usually focuses on differently methylated regions (DMRs) identified by comparing methylation profiles of several samples divided into two different groups. An example would be the comparison of tumor samples with blood samples. There are different approaches and tools to detect DMRs, e.g. `bsmooth` (Hansen et al., 2012) – one of the most popular DMR callers. It identifies differentially methylated CpGs (DMCs) by t-test statistics and connects adjacent DMCs to larger regions (DMRs). `Camel` includes an implementation of this algorithm:

Let S be a set of samples, $X_s \in \{0, 1\}$ the class of sample $s \in S$, $S_i = \{s \in S : X_s = i\}$ the set of class i specific samples, $n_i = |S_i|$ the number of class i specific samples and $\beta_s(c) \in [0, 1]$ the β-values of CpG c in sample s. We

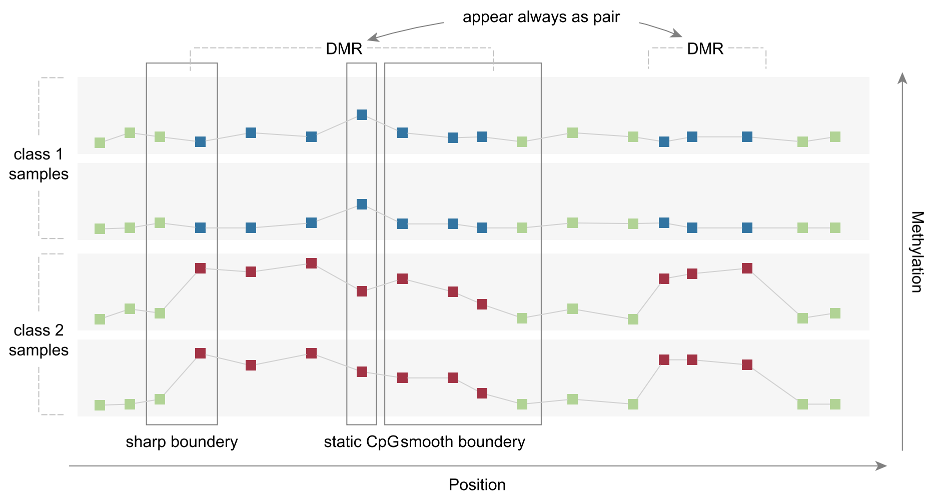


FIGURE 3.8: Two DMRs over two classes, each with two samples and different annotated properties. Boxes represent CpG with background methylation (green), high methylation (red) and low methylation (blue) in relation to mean methylation. We call the left boundary of the left DMR *sharp* because the methylation of the first DMC encountered is already on the DMR mean level. On the right side of the same DMR, we illustrate a *smooth* boundary at which the degree of methylation gently decreases until it reaches the background level. We call a CpG within a DMR static if there is no methylation difference between the sample classes. So this CpG is not a DMC. In some cases, a second, slightly smaller DMR can be observed, whose appearance is bound to the first.

build a t-test statistic

$$t(c) = \frac{\mu_1(c) - \mu_0(c)}{\sigma(c)\sqrt{\frac{1}{n_0} + \frac{1}{n_1}}} \quad \text{where} \quad \mu_i(c) = \frac{1}{n_i}\sum_{s \in S_i} \beta_s(c)$$

with average group methylation difference $\mu_1(c) - \mu_0(c)$ and standard deviation for the methylation values

$$\sigma(c) = \sqrt{\frac{1}{n_0}\sum_{s \in S_0}(\beta_s(c) - \mu_0(c))^2 + \frac{1}{n_1}\sum_{s \in S_1}(\beta_s(c) - \mu_1(c))^2}$$

of all samples at a given c. Neighboured CpGs with $t(c) > 0 > q^+_{0.95}$ are then merged to hypermethylated and $t(c) < 0 < q^-_{0.05}$ to hypomethylated DMRs, where $q^+_{0.95}$ is the (adjustable) 95% quantile of all positive t and $q^-_{0.05}$ the (adjustable) 5% quantile of all negative t.

Finally, the method filters previously identified regions according to a minimum number of CpGs and the mean DMR group methylation difference Δ. Through several experiments (see Chapter 7) we were able to observe variable properties of DMRs, especially at their borders. This is visualized in Figure 3.8.

- Some DMRs have *sharp* boundaries, where the methylation value changes rapidly on the first CpG encountered. Others increase or decrease *smoothly* over several CpGs up to their final level. `Bsmooth`'s parameters, such as the quantile threshold and smoothing parameters (see below), have a massive impact on which CpG become part of the DMR and which CpG is selected as background methylation.

- Some DMRs overspan CpGs with normal background methylation. These could be considered as two distinct DMRs separated by non-DMCs. Since they always appear in pairs, they are comparable to haplotype SNPs. Depending on calling algorithm and parameters, they are often called single big continuous DMRs rather than two separated.
- Some DMRs contain *static* CpGs, that have the same methylation level in every sample. These CpGs could lower or raise the DMRs methylation level significantly, especially if the DMR consists only of a few CpGs.

Based on these observations, we have modified our implementation.

`Bsmooth` has initially been developed to handle data with low coverage. For this reason, `bsmooth` applies sample-wise smoothing pre-processing to the methylation profile to compensate for errors in the methylation measurement due to too small coverage and thus increase the DMR detection rate. In recent years, however, coverage of WGBS data has increased to a point where smoothing can be considered unnecessary. This not only makes smoothing superfluous but also detrimental, as it leads to blurred boundaries. This makes accurate boundary prediction difficult, even if it is clearly defined. Although disabled by default, the DMR call algorithm of `camel` still offers the possibility to smooth the input with a Savitzky-Golay filter (Savitzky and Golay, 1964), which itself performs a polynomial regression (window size = 10 CpGs, polynomial degree = 3).

We have addressed the problems mentioned above of the different DMR peculiarities found in the following way. After merging adjacent DMCs to build DMRs, our algorithm once again merges nearby DMRs separated at most by k non-DMCs (default $k = 3$). This preserves sharp boundaries, yet allows static CpGs in each DMR, and provides a way to configure DMR separation/unification. However, this can cause multiple non-DMCs in a DMR to distort the mean methylation group difference. This, in turn, becomes problematic for the final step, removing DMRs with low mean differences.

Unlike `bsmooth`, which uses the simple average, we build a weighted average methylation. $\mu_s^*(d) = \frac{1}{|C(d)|}\sum_{c \in C(d)} \sigma(c) \cdot \beta_s(c)$ for each DMR d, its set of CpGs $C(d)$ and sample s. We call $\mu_s^*(d)$ the *core methylation* of DMR d for the sample s. It improves the accuracy of the group methylation by weighting the influence of a CpG on the mean methylation with its standard deviation. CpGs with static or background methylation values that do not differ between samples at all are ignored and split DMRs become less problematic. This allows our algorithm, represented in Algorithm 4, to address these difficulties without the drawbacks a smoothing process.

```
Data: case samples, control samples, a list of chromosomes
Result: Differentially methylated regions
Function CallDMRs(List<Samples> case, List<Samples> control, List<Int>
chromosomes)
    // Create a dictionary for the t-stats
    t = dictionary
    foreach chrom in chromosome do
        // optional smoothing
        CpGc = smooth_values(chrom)
        foreach c in CpGc do
            // calculate t-stat for each CpG
            t[c] = calculate_t_stat(c, case, control)

    // get all positive t-stat values
    t_pos = all t > 0
    // get the 95% quantile for the positive t
    threshold_pos = quantile(t_pos, 0.95)
    // get CpGs with high positive t-stat > threshold
    c_selected_pos = all c with t[c] > threshold_pos
    // merge close CpGs to DMRs
    dmrs_pos = merge(c_selected_pos, maxdist = 3)


    // repeat for low negative t-stats
    t_neg = all t < 0
    threshold_neg = quantile(t_neg, 0.05)
    c_selected_neg = all c with t[c] < threshold_neg
    dmrs_neg = merge(c_selected_neg, maxdist = 3)


    // unify DMRs
    dmrs = dmrs_pos + dmrs_neg
    // filter DMRs
    dmrs = filter(dmrs, delta = 0.3, min_cpg_count = 4)
    return dmrs
```

ALGORITHM 4: Algoritm for calling DMRs.

3.4 Downstream Analysis

Storing *index-file* and *camel-data-file* in hdf5 has advantages. Thanks to the hierarchical structure, data can also be efficiently retrieved, modified and processed using tools or scripts written in other languages due to the various hdf5 parsing libraries available (see Section 1.7).

In addition to its core functions - calling the CpG methylation level and

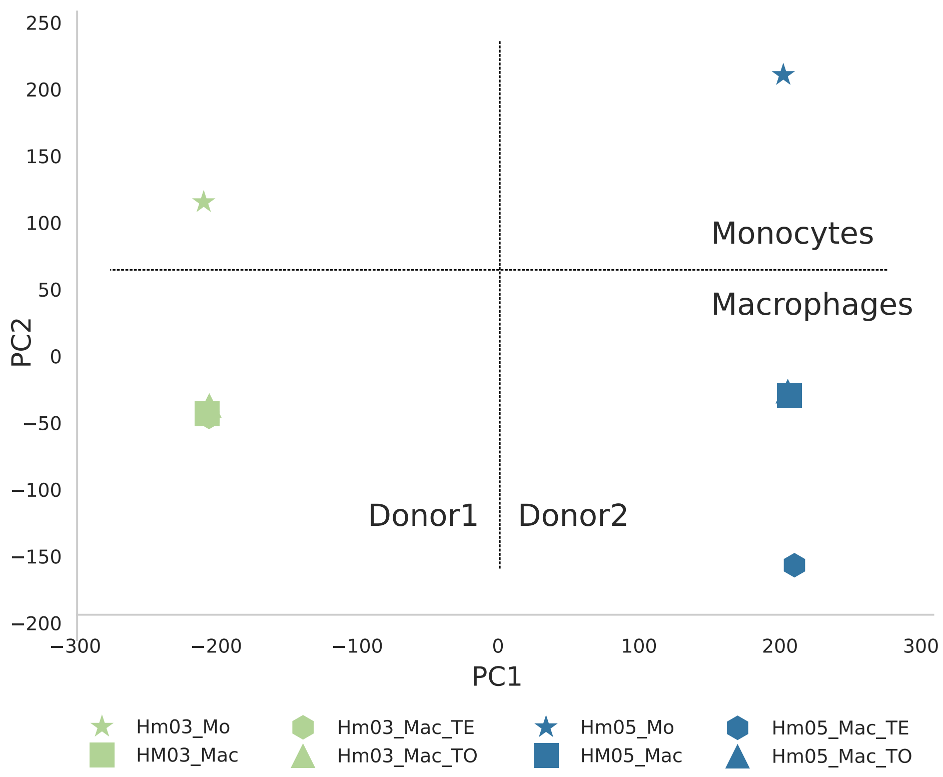


FIGURE 3.9: Example PCA of eight samples of monocytes and macrophages (three different conditions) from two different donors. As expected, PCA shows a clear separation between donors and cell types.

identifying differentiated regions - `camel` itself provides several downstream functions to analyze or visualize the previously determined methylation levels. All of them are based on the *camel-index* and the corresponding *camel-data-files*. The simplest way to display the data is in the form of text. Other tools have already established different text formats for input and output. For compatibility reasons, `camel` can output its data in the most common formats, e.g., the coverage output format of Bismark, the output format generated by Bis-SNP or a native simplified bed format. This variety of formats allows the data to be visualized with the Integrative Genomics Viewer (IGV) (Thorvaldsdóttir et al., 2013) or processed with external tools that support one of the standard formats, e.g., the widely used RnBeads (Assenov et al., 2014).

In addition to the text, `camel` offers functions for graphical output. It can perform a principal component analysis (PCA), t-distributed stochastic neighborhood embedding (t-SNE) (Maaten and Hinton, 2008) and clustering of the 1000 most variant CpGs to visualize the general similarity between the samples. Figure 3.9 shows an example of a PCA from Chapter 7 (Figure 7.1) of four samples of four different diseases from two donors. Details on this figure can be found in the Chapter 7.

We have also implemented a nucleosome protection plot for NOMe-seq, which can be seen in Figure 3.10. In this diagram type, NOMe-seq-induced GpC methylation is calculated as a function of distance to a transcription start point. A typical wave pattern is shown which can react to different

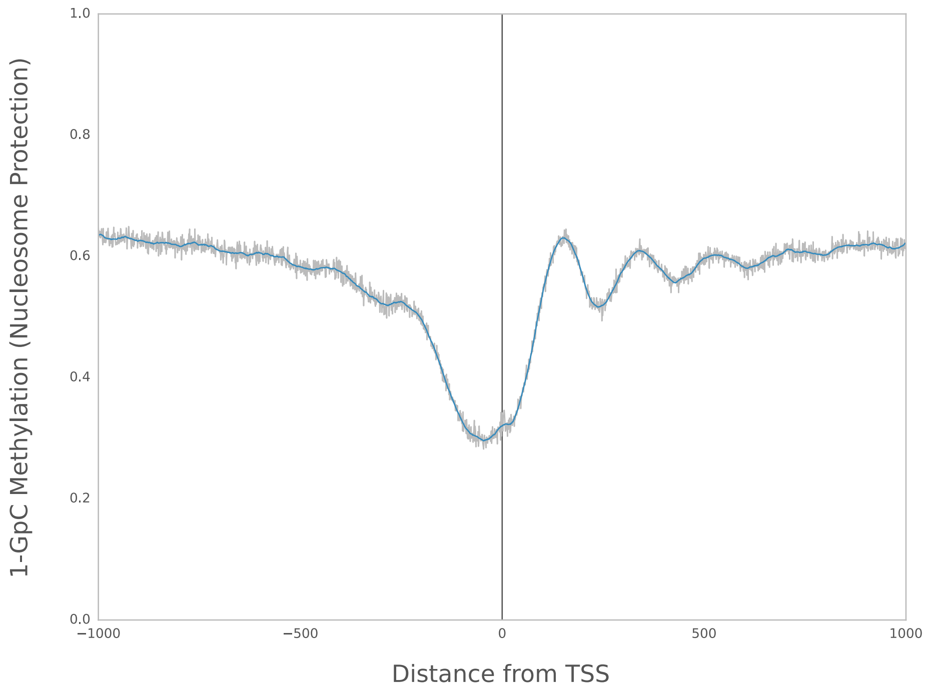


FIGURE 3.10: Example of a nucleosome protection plot. The mean GpC methylation within the CpG islands relative to all transcription start sites (-1000 bp to 1000 bp) shows the typical wave pattern. Different cell conditions and diseases can alter this pattern and indicate transcription problems that are the subject of current research. Biological interpretations of each partial wave go beyond the scope of this work.

cell conditions with changes.

Furthermore `Camel` provides some useful helper functions. Be S a set of *camel-data-files*, I the corresponding index and D a set of regions, for example DMRs. It can generate synthetic average, maximum and minimum samples of S and calculate the core methylation for each sample in S for each region in D. The provision of I also makes it possible to simulate regions similar in size to the given regions in D, with a minimum number in CpG. We used all these functions for example in the study described in Chapter 7.

3.5 Software

`Camel` has been implemented in Python 3, while all computationally expensive parts are realized using *numpy* and *scipy* functions. `Camel` is open source (MIT license) and may be obtained from its BitBucket code repository[1]. We recommend to follow the README instructions at the repository for easy installation and the user-manual at gitbook[2]. No commercial or proprietary optimization libraries are required.

[1]https://bitbucket.org/christopherschroeder/camel
[2]https://www.gitbook.com/book/christopherschroeder/camel-documentation

3.6 Evaluation

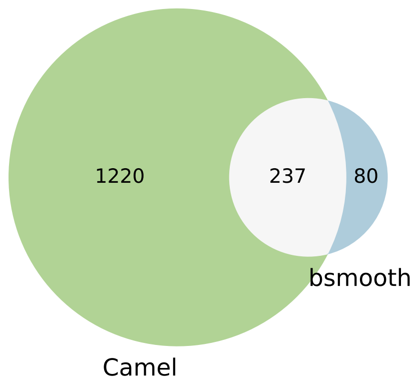
For newly developed methods, a geFor newly developed methods, a general evaluation strategy is to apply this either to simulated data or to real data with the known result. The calculated result is then compared with the ground truth to calculate the true and false positive and negative rates. Different parameter settings result in different precision and recall values, which are then summarized in a ROC curve (Hajian-Tilaki, 2013). In the case of DMR detection, this requires WBGS methylomes of several biological replicates of at least two different groups with known differential methylation. However, in contrast to genetic variations in diploid organisms, where the differences are usually found in 50% (heterozygous variant) or 100% (homozygous variant) of the data, methylation differences between samples can be very small, and the biological variance between samples in a group causes additional noise. Many biological replicates with high coverage are required to distinguish even small differences from noise Besides, all samples must be sequenced by the same institute and sequencing protocols to avoid bias (see Chapter 7). For improved evaluation, several samples from several different groups should be compared, as different tissue comparisons may lead to DMRs of different size and shape. Due to high sequencing costs, such data are not available, and there is not enough information about biological differences and character traits of DMRs to model a suitable simulator, even if this would be interesting and useful.

Instead, it is common to evaluate the newly developed method by comparing its DMRs with existing approaches. For this purpose, we use data from DEEP[3], the German epigenome program focusing on the production and interpretation of 80 reference genomes of selected human cells and tissues. The data consist of monocytes and macrophages from two donors. Since our method is a modification of `bsmooth`, we compare our tool with the original approach. Chapter 7 also deals with these monocytes to perform an interindividual methylation analysis of several independent samples.

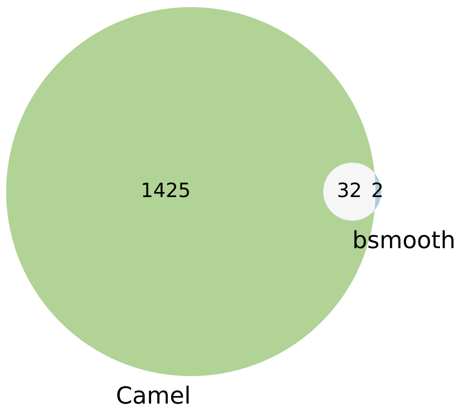
We configured `camel` to call DMRs with realistic parameters of a minimum DMR methylation difference $\Delta = 0.4$ and a minimum size of $n = 4$ CpGs and a maximum number of undifferentiated CpGs of 3, as used for real data. These DMRs are then used for pairwise comparison with DMRs calculated by `bsmooth` and `metilene` under different settings.

The smoothing step of `bsmooth` usually reduces the methylation difference. We have therefore decided to reduce the difference from `camel` to $\Delta = 0.3$. A close examination of the resulting regions with these parameters reveals inaccurate boundaries and thus an incorrect number of CpGs and a lower methylation difference for most DMRs. This leads to low detection rates and is caused by smoothing parameters optimized for low coverage. To improve the fairness of the rating, we have reduced the calculation with the smoothing of the genomic window size w from 1000 bp (default) to 500 bp, and the minimum amount of windowed CpGs h from 70 (default) to 35.

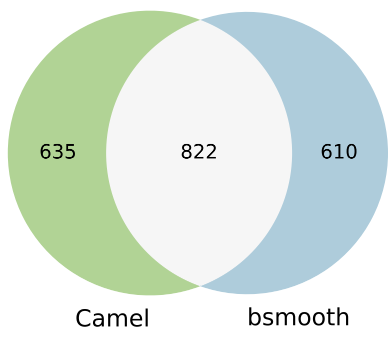
[3]http://epigenomesportal.ca/ihec/



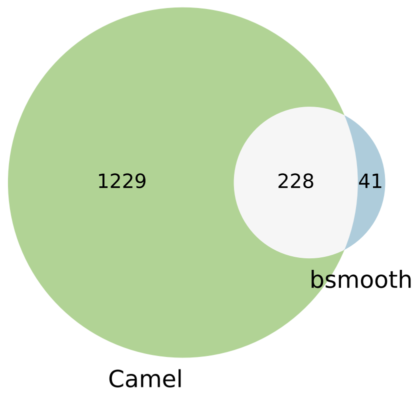
(a) bsmooth
$h = 70$, $w = 1000$, $n = 4$, $\Delta = 0.2$



(b) bsmooth
$h = 70$, $w = 1000$, $n = 4$, $\Delta = 0.3$



(c) bsmooth
$h = 35$, $w = 500$, $n = 4$, $\Delta = 0.2$



(d) bsmooth
$h = 35$, $w = 500$, $n = 4$, $\Delta = 0.3$

FIGURE 3.11: Comparison of DMRs called by `Camel` against `bsmooth` with different parameters. `Camel` is set to a minimum DMR size of $n = 4$ CPGs and difference $\Delta = 0.4$. The parameters of `bsmooth` are specified in each subfigure.

This adjustment reduces the total number of CpGs considered for smoothing and, as expected, results in a larger number of precise DMRs. We call a DMR *exclusive* if there is no overlap with a DMR called the other tool. Figure 3.11 (a to d) shows four Venn diagrams of DMRs called by `camel` and `bsmooth` for different parameter configurations. We examined the exclusive DMRs of `camel`. To investigate only highly secure `camel`-exclusive DMRs, we further lowered Δ for `bsmooth` to 0.2 to exclude DMRs that were not even recognized by unrealistically sensitive parameters, as shown in Figure 3.11 (c). We then manually examined randomly selected ranges of 635 `camel`-exclusive DMRs. All regions are optically convincing, even if there are only small differences.

On the other hand, we also examined randomly selected DMRs of the 43 `bsmooth`-exclusive regions shown in Figure 3.11. (b and d). In all DMRs examined, we could either see no difference or, if so, only a weak one in three

CpGs or less. These small areas are not displayed (correctly) by `camel`, because they fall below the selected minimum size of four CpGs. We assume that the smoothing step is responsible for this behavior and conclude that our modifications have a real (positive) influence on the detection rate and the false positive rate.

3.7 Summary

In this chapter, we present the two approaches to aligning bisulfite converted reads to a standard reference genome: Wildcard Alignment and Three-Letter Alignment. We also show how three-letter alignment differs between 32-bit and 64-bit references and how this basic concept allows any generic read mapper to align WGBS data. Besides, we have developed `camel`, a caller for methylation levels for bisulfite converted reads that, unlike others, only recognizes and stores the degree of methylation at specific reference positions. This not only allows you to ignore most data (only 1% of base pairs in the human genome are CpGs) but also simplifies the downstream analysis, since each sample contains an equal number of reads at corresponding positions. Among other useful functions, such as PCA or the calculation of an average sample, `Camel` also includes a DMR calling algorithm. Our implementation and changes to the algorithm of the widely used `bsmooth` based on the discovered DMR properties also lead to improved DMR detection.

Chapter 4

Beta Distribution Fitting

When working with methylation, it is sometimes useful to consider the distribution of several measured methylation levels, which can be described by a beta mixture model. An example is given in Section 7.3, where we deal with differentially methylated regions between individuals and use such distributions among other things to identify DMR related SNPs.

The beta distribution is a continuous probability distribution that assumes values in the unit interval $[0, 1]$. In addition to the degree of DNA methylation of CpG dinucleotides or average methylation of larger genomic regions, it has been used in various bioinformatics applications to model data that naturally assume values between $0 and 1$, such as relative frequencies, probabilities, or absolute correlation coefficients. One of the best-known applications is the estimation of false discovery rates (FDRs) from p-value distributions after several tests by adaptation of a beta-uniform mixture (BUM, Pounds and Morris, 2003). By linear scaling, beta-mixture distributions are able to model any set whose values come from a finite interval $[L, U] \subset \mathbb{R}$.

This leads to the problem of estimating these scaling parameters and the actual parameters for the individual beta distributions in the mixture model. A popular and successful method for parameter optimization in mixtures is the expectation maximization (EM) algorithm (Dempster et al., 1977), which is based on *Maximum Likelihood*. (ML) Estimation (MLE). It iteratively solves an ML problem for each estimated component and then re-estimates which data points belong to which component. However, as shown in Section 4.2, MLE is not suitable for beta distributions. The main problem is that the probability function may not be finite (for some parameter values) if some of the observed data points are 0 or 1, as often observed in methylation levels. Since MLE is used for a *single* beta distribution is problematic, the use of EM for beta mixtures is even more difficult - unless ad hoc corrections are made. In this chapter, we propose an algorithm for parameter estimation in beta mixtures called *iterated method of moments* published in (Schröder and Rahmann, 2017).

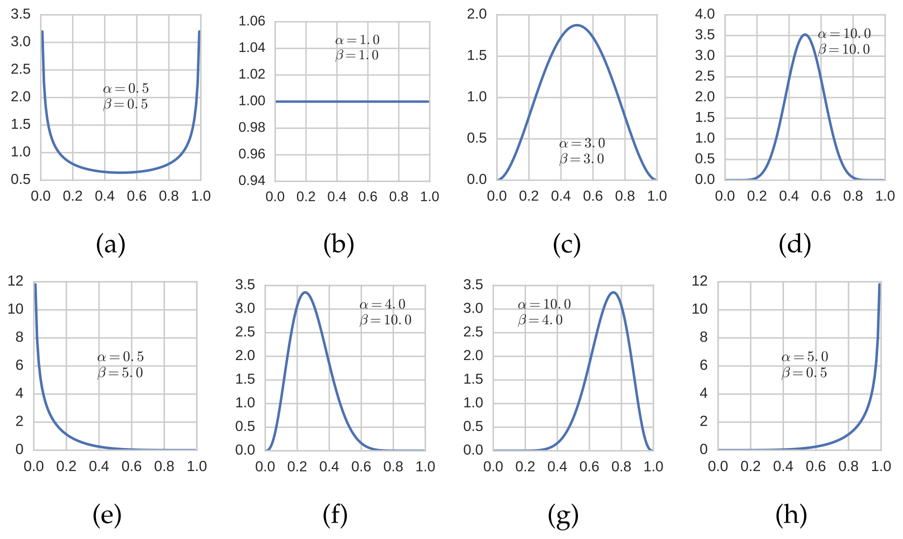


FIGURE 4.1: Different shapes of beta distributions depending its shape parameters α and β.

4.1 The Beta Distribution

The beta distribution with parameters $\alpha > 0$ and $\beta > 0$ is a continuous probability distribution on the unit interval $[0, 1]$ whose density on $[0, 1]$ is given by

$$b_{\alpha,\beta}(x) = \frac{1}{B(\alpha, \beta)} \cdot x^{\alpha-1} \cdot (1-x)^{\beta-1}, \qquad \text{where } B(\alpha, \beta) = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)},$$

and Γ refers to the gamma function $\Gamma(z) = \int_0^\infty x^{z-1}\,\mathrm{e}^{-x}\,\mathrm{d}x$ with $\Gamma(n) = (n-1)!$ for positive integers n. It can be shown that $\int_0^1 b_{\alpha,\beta}(x)\,\mathrm{d}x = 1$.

The distribution can take a variety of shapes depending on whether $0 < \alpha < 1$ or $\alpha = 1$ or $\alpha > 1$ and $0 < \beta < 1$ or $\beta = 1$ or $\beta > 1$; see Figure 4.1. Note that for $\alpha = \beta = 1$, we obtain the uniform distribution.

If X is a random variable with a beta distribution, then its expected value μ and variance σ^2 are

$$\mu := \mathbb{E}[X] = \frac{\alpha}{\alpha+\beta}, \qquad \sigma^2 := \mathrm{Var}[X] = \frac{\mu(1-\mu)}{\alpha+\beta+1} = \frac{\mu(1-\mu)}{1+\phi},$$

where $\phi = \alpha + \beta$ is often called a *precision* parameter; large values indicate that the distribution is concentrated.

Conversely, the parameters α and β may be directly expressed in terms of μ and σ^2. First, compute

$$\phi = \frac{\mu(1-\mu)}{\sigma^2} - 1; \qquad \text{then} \quad \alpha = \mu\phi, \quad \beta = (1-\mu)\phi. \tag{4.1}$$

While a single beta distribution can take a variety of shapes, *mixtures* of beta distributions are even more flexible. Such a mixture has the general form

$$f_\theta(x) = \sum_{j=1}^{c} \pi_j \cdot b_{\alpha_j,\beta_j}(x)\,, \tag{4.2}$$

where c is the *number of components*, π_j are *mixture coefficients* satisfying $\sum_j \pi_j = 1$ and $\pi_j \geq 0$, and α_j, β_j are called *component parameters*. In total, we refer to all of these as *model parameters* and abbreviate them as θ. The number of components c is often assumed to be a given constant and not part of the parameters to be estimated.

4.2 Maximum Likelihood Estimation for Beta Distributions

Maximum likelihood (ML) estimation (MLE) is a frequently used paradigm for estimating θ from n observed usually independent samples $(x_1, \dots, x_n)$ such that the observations are well explained by the resulting distribution (the *parameter estimation problem*). This can be expressed as an optimization problem, as follows:

$$\text{Given } (x_1,\dots,x_n),\ \ \text{maximize } \mathcal{L}(\theta) := \prod_{i=1}^{n} f_\theta(x_i),$$
$$\text{or equivalently, } L(\theta) := \sum_{i=1}^{n} \ln f_\theta(x_i).$$

Since θ represents the parameters and $f_\theta(x)$ is the probability density of a single observation, the goal is to find parameters θ^* that maximize $L(\theta)$.

Because $\gamma(y) := \ln\Gamma(y)$, the beta log-likelihood can be written as

$$L(\alpha,\beta) = n(\gamma(\alpha+\beta)-\gamma(\alpha)-\gamma(\beta))+(\alpha-1)\cdot\sum_i \ln x_i+(\beta-1)\cdot\sum_i \ln(1-x_i)\,.$$

The optimum conditions $\mathrm{d}L/\mathrm{d}\alpha = 0$ and $\mathrm{d}L/\mathrm{d}\beta = 0$ are solved numerically and iteratively. While this is inconvenient (in comparison to a mixture of Gaussians where analytical formulas exist for the ML estimators), the main problem is a different one. The log-likelihood function is not well defined for $\alpha \neq 1$ or $\beta \neq 1$ if any of the observations x_i are 0 or 1. Several implementations of ML estimators for beta distributions (e.g. `betareg`, see below) fail with error in these cases.

Note that there is no problem *in theory*, because $x_i \in \{0, 1\}$ is an event of probability zero if the data are truly generated by a beta distribution. Real data – such as observed methylation levels – may very well take these values. The primary motivation of this chapter is to work with beta mixtures and observations of $x = 0$ and $x = 1$ in a principled way.

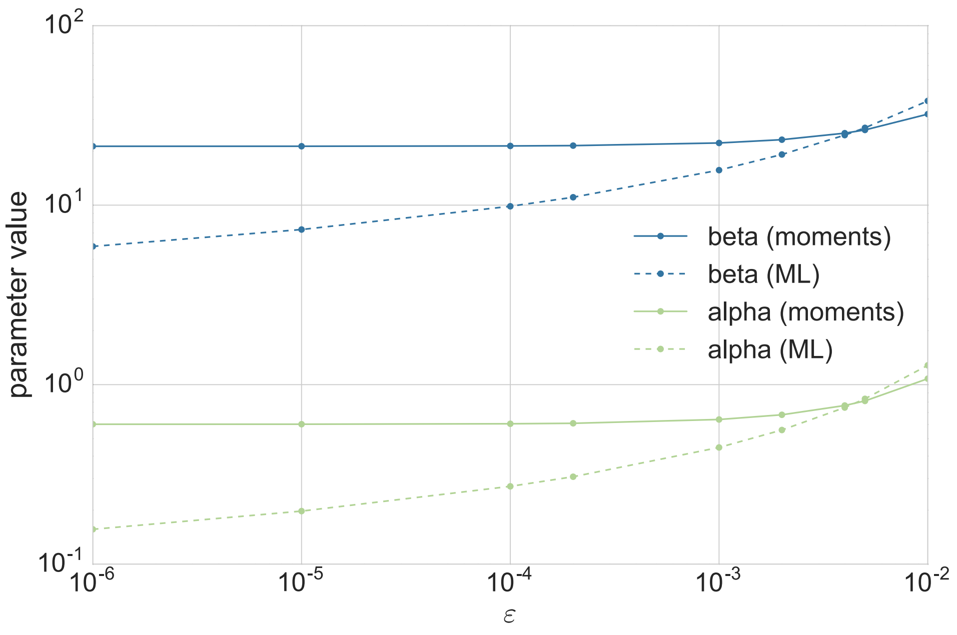


FIGURE 4.2: Estimated parameter values α (green) and β (blue) from a dataset consisting of the ten observations $0.01, \ldots, 0.10$ and ten observations of ε for varying values of ε. Estimation was done using MLE (dotted lines) as implemented in the R package `betareg` and by our (moment-based) method (solid lines).

A typical ad-hoc solution is to linearly rescale the unit interval $[0, 1]$ to a smaller sub-interval $[\varepsilon, 1 - \varepsilon]$ for some small $\varepsilon > 0$ or to simply replace values $< \varepsilon$ by ε and values $> 1 - \varepsilon$ by $1 - \varepsilon$, such that, in both cases, the resulting adjusted observations are in $[\varepsilon, 1 - \varepsilon]$.

A simple example shows that the resulting parameter estimates depend strongly on the choice of ε in the ML paradigm. Consider 20 observations, half of them at $x = 0$, the other half at $x = 0.01, \ldots, 0.10$. For different values of $0 < \varepsilon < 0.01$, replace the ten zeros by ε and compute the ML estimates of α and β. We used the R package `betareg`[1] (Grün et al., 2012), which performs numerical ML estimation of $\text{logit}(\mu)$ and $\ln(\phi)$, where $\text{logit}(\mu) = \ln(\mu/(1 - \mu))$. We then used Equation 4.1 to compute ML estimates of α and β. We additionally used our new approach (Section 4.4) with the same varying ε. In contrast to traditional MLE, our approach also works with $\varepsilon = 0$. The resulting estimates for α and β are shown in Fig. 4.2: Not only is our approach able to use $\varepsilon = 0$ directly; it is also insensitive to the choice of ε for small $\varepsilon > 0$.

In summary, MLE is known to be statistically efficient for correct data, but its results can be sensitive to data corruption. The problems of MLE are particularly severe when modeling with beta distributions. The probability function is not well defined for reasonable real data sets, and the solution depends heavily on ad hoc parameters. Before we present our solution to these problems, we first discuss parameter estimation in mixed models.

[1] `https://cran.r-project.org/web/packages/betareg/betareg.pdf`

4.3 The EM Algorithm for Beta Mixture Distributions

For parameters θ of mixture models, including each component's parameters and mixture coefficients, the log-likelihood function

$$L(\theta) = \sum_{i=1}^{n} \ln f_\theta(x_i) \quad \text{with } f_\theta(x_i) \text{ as in Eq. (4.2)}$$

often has many *local* maxima. Also a global optimal solution is difficult to compute.

The EM algorithm (Dempster et al., 1977) is a general iterative method for ML parameter estimation for incomplete data. In mixed models, the *missing* data is the information which data point belongs to which component. However, this information can be estimated in the E-step (expectation step) (for initial parameter estimates) and then used to derive better parameter estimates of ML for each component separately in the M-step (maximization step). In general, EM slowly converges to a local optimum of the log-likelihood function (Redner and Walker, 1984).

E-Step To estimate the expected responsibility $W_{i,j}$ of each component j for each data point x_i, the component's relative probability at that data point is computed, such that $\sum_j W_{i,j} = 1$ for all i.

$$W_{i,j} = \frac{\pi_j \, b_{\alpha_j,\beta_j}(x_i)}{\sum_k \pi_k \, b_{\alpha_k,\beta_k}(x_i)} \quad \text{and} \quad \pi_j^+ = \frac{1}{n}\sum_{i=1}^{n} W_{i,j}\,. \tag{4.3}$$

M-Step. By using the responsibility weights $W_{i,j}$, the components are unmixed, and a separate (weighted) sample is obtained for each component so that its parameters can be estimated independently of each other using MLE. The ML estimates of the new mixing coefficients π_j^+ in equation 4.3 are the average values of the responsibility weights across all samples.

Initialization and termination. EM requires initial parameters before starting an E-step. The resulting probable local optimum depends on these initial parameters. Therefore, it is common to either select the initial parameters based on additional information (e.g., a component with small values, a component with large values) or to restart EM with different random initializations. Convergence is detected by monitoring relative changes in logarithmic probability or parameters between iterations and stopping if these changes are below a specified tolerance.

Properties and problems with beta mixtures. One of the main reasons why the EM algorithm is mainly used in practice for mixture estimation is the availability of an objective function (the log-likelihood). Due to Jensen's inequality, it increases with each EM iteration until it reaches a stationary point (Redner and Walker, 1984). Someone can objectively compare solutions obtained by two runs with different initializations, based on their log-likelihood values.

In beta mixtures, there are several problems with the use of the EM algorithm. First, the responsibility weights $W_{i,j}$ for $x_i = 0$ or $x_i = 1$ are not well defined because of the singularities in the likelihood function described in Section 4.1. Second, the M step cannot be executed if the data contains such a point for the same reason. Third, even if all $x_i \in]0, 1[$, the resulting mixtures are sensitive to data corruption. Fourth, since each M-step already contains a numerical iterative maximization, the arithmetic load over multiple EM iterations is significant. In the following, we propose our algorithm for parameter estimation in beta mixtures, which does not suffer from these drawbacks.

4.4 The Iterated Method of Moments

With the necessary preliminaries in place, the basic idea of the algorithm can be stated compactly.

From the initial parameters, proceed iteratively as in the EM frame and alternate between an E-step, which is a small modification of the E-step of EM, and a parameter estimation step, which is not based on the ML paradigm, but Pearson's moment method, until a stationary point is reached.

To estimate Q free parameters, the approach is to choose Q moments of the distribution, express them through the parameters, and equate them with the corresponding Q moment values. This is usually equivalent to solving a system of nonlinear Q equations. The method of moments is applied directly to the mixture distributions. For example, a mixture of two one-dimensional Gaussian equations has five free parameters: two mean values μ_1, μ_2, two variances σ_1^2, σ_2^2, and the weight π_1 of the first component. Thus one needs to choose five moments, say $m_k := \mathbb{E}[X^k]$ for $k = 1, \ldots, 5$ and solve the corresponding relations. The solution of these equations for many components (or in high dimensions) seems frightening, even numerical. Moreover, it is not clear whether there is always a unique solution.

However, for a single beta distribution, α and β are easily estimated from the sample mean and variance by equation 4.1 using sample moments instead of real values. To avoid the problems of MLE in beta distributions, we replace the Likelihood Maximization Step (M-Step) in EM by a Method of Moments Estimation Step (MM-Step) with expectation and variance.

We, therefore, combine the idea of using latent responsibility weights of EM with the instantaneous estimation but avoid the problems of pure instantaneous estimation (large nonlinear systems of equations).

Initialization

A reasonable general strategy for beta mixtures is to let each component concentrate on a specific sub-interval of the unit interval. For c components, we start with a component that is responsible for values around $k/(c-1)$ for each $k = 0, \dots, c-1$. The expectation and variance of the near $k/(c-1)$ component is initially first from the corresponding sampling moments of all data points in the $[(k-1)/(c-1), (k+1)/(c-1)] \cap [0,1]$ interval. (If an interval contains no data, the component is removed from the model.) The initial mixing coefficients are estimated in proportion to the number of data points in that interval.

E-step

The E-step is essentially the same as for EM, except that we assign weights explicitly to data points $x_i = 0$ and $x_i = 1$.

Let j_0 be the component index j with the smallest α_j. If there is more than one, choose the one with the largest β_j. The j_0 component takes full responsibility for all i with $x_i = 0$, i.e., $W_{i,j_0} = 1$ and $W_{i,j} = 0$ for $j \neq j_0$. Similarly, let j_1 be the component index j with the smallest β_j (among several ones, the one with the largest α_j). For all i with $x_i = 1$, set $W_{i,j_1} = 1$ and $W_{i,j} = 0$ for $j \neq j_1$.

MM-step

The MM-step estimates mean and variance of each component j by the responsibility-weighted sample moments,

$$\mu_j = \frac{\sum_{i=1}^n W_{ij} \cdot x_i}{\sum_{i=1}^n W_{ij}} = \frac{\sum_{i=1}^n W_{ij} \cdot x_i}{n \cdot \pi_j}, \tag{4.4}$$

$$\sigma_j^2 = \frac{\sum_{i=1}^n W_{ij} \cdot (x_i - \mu_j)^2}{n \cdot \pi_j}.$$

Then α_j and β_j are computed according to Equation 4.1 and new mixture coefficients according to Equation 4.3.

Termination

Let θ_q be any real-valued parameter to be estimated and T_q a given threshold for θ_q. After each MM-step, we compare θ_q (old value) and θ_q^+ (updated value) by the relative change $\kappa_q := |\theta_q^+ - \theta_q| \,/\, \max\left(|\theta_q^+|, |\theta_q|\right)$. (If $\theta_q^+ = \theta_q = 0$, we set $\kappa_q := 0$.) We say that θ_q is stationary if $\kappa_q < T_q$. The algorithm terminates when all parameters are stationary.

Properties

The proposed method has no objective function that can be maximized. Therefore we can neither make statements about the improvement of such a function nor directly compare two solutions from different initializations by objective function values. It also makes no sense to talk about "local optima", but analogous to the EM algorithm there can be several stationary points. We have not yet determined whether the method always converges. On the other hand, we have the following desirable property. In each MM-step, before updating the component weights, the expectation of the estimated density equals the mean value of the sample. This is especially true at a stationary point.

For a density f we write $\mathbb{E}[f]$ for its expectation $\int x \cdot f(x)\,\mathrm{d}x$. For the mixture density Equation 4.2, we have by linearity of expectation that $\mathbb{E}[f_\theta] = \sum_j \pi_j\,\mathbb{E}[b_{\alpha_j,\beta_j}] = \sum_j \pi_j\,\mu_j$. Using Equation 4.4 for μ_j, this is equal to $\frac{1}{n}\sum_j \sum_i W_{ij}\,x_i = \frac{1}{n}\sum_i x_i$, because $\sum_j W_{ij} = 1$ for each j. Thus $\mathbb{E}[f_\theta]$ equals the sample mean. □

4.5 Evaluation

We evaluated our method from two points of view: the determined class allocation against the true component origin of the data points and the number of estimated components against the actual number.

4.5.1 Simulation and Fitting for Class Assignment

We investigate the advantages of beta-mixture modeling by simulation. In the following, let U be a uniform random number from $]0,1[$. We generate two datasets, each consisting of 1000 three-component mixtures. The first (second) dataset consists of 200 (1000) samples per mixture.

To generate a mixture model, we first pick mixture coefficients $\pi = (\pi_1, \pi_2, \pi_3)$ by drawing U_1, U_2, U_3, computing $s := \sum_j U_j$ and setting $\pi_j := U_j/s$. This does not generate a uniform element of the probability simplex but induces a bias towards distributions where all components have similar coefficients, which is reasonable for the intended application.

The first component represents the unmethylated state; therefore we choose an $\alpha \leq 1$ and a $\beta > 1$ by drawing U_1, U_2 and setting $\alpha := U_1$ and $\beta := 1/U_2$. The third component represents the fully methylated state and is generated symmetrically to the first one. The second component represents the semi-methylated state (0.5) and should have large enough approximately equal α and β. We draw U_1, U_2 and define $\gamma := 5/\min\{U_1, U_2\}$. We draw V uniformly between 0.9 and 1.1 and set $\alpha := \gamma V$ and $\beta := \gamma/V$.

To draw a single random sample x from a mixture distribution, we first draw component j according to π and then a value x from the beta distribution with parameters α_j and β_j. After drawing $n = 200$ (dataset 1) or $n = 1000$ (dataset 2) samples, we modify the result as follows. For each mixture sample from dataset 1, we set the three smallest values to 0.0 and

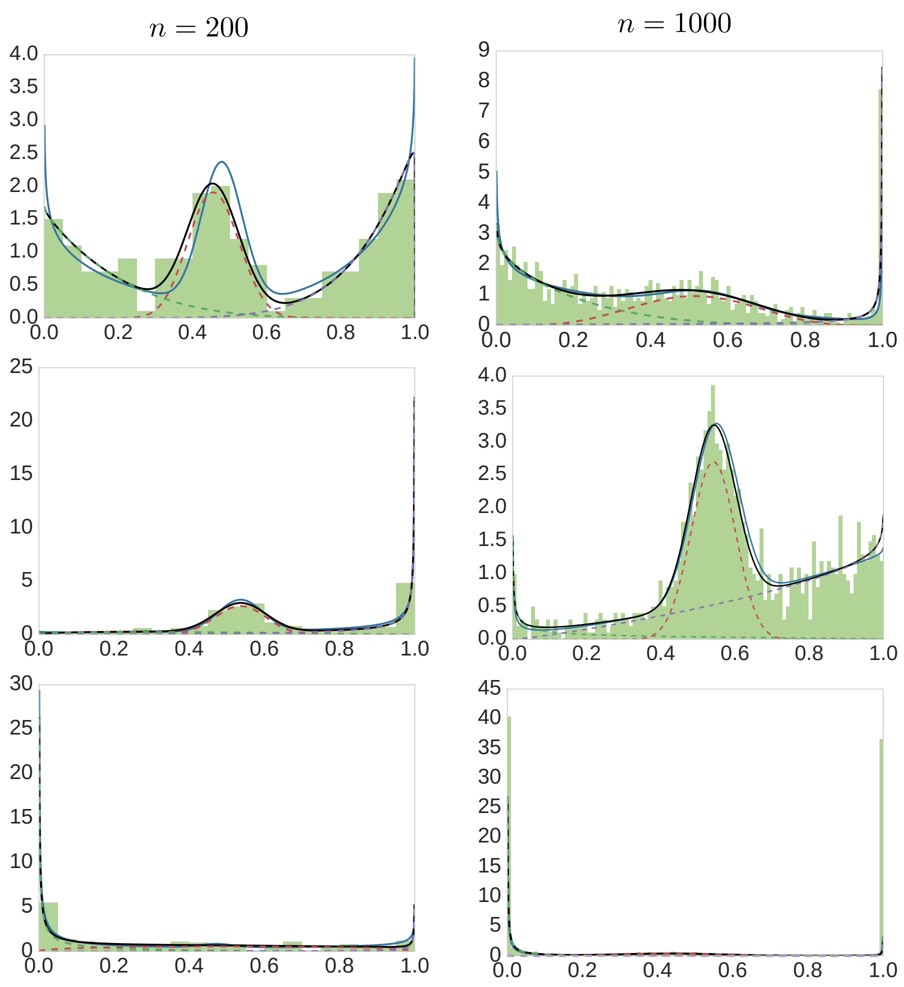


FIGURE 4.3: Examples of generated three-compoment beta mixtures(blue solid lines), data samples (light green histograms) and fitted mixture models (black solid lines). Dashed lines show estimated weighted component densities (green: unmethylated; red: semi-methylated; magenta: fully methylated). Left row: examples with $n = 200$ samples; right row: $n = 1000$.

the three largest values to 1.0. In dataset 2, we proceed similarly with the ten smallest and largest values.

We use the algorithm described in Section 4.4 to fit a mixture model with a slightly different initialization. The first component is estimated from the samples in $[0, 0.25]$, the second one from the samples in $[0.25, 0.75]$ and the third one from the samples in $[0.75, 1]$. The first (last) component is enforced to be falling (rising) by setting $\alpha_1 = 0.8$ ($\beta_3 = 0.8$) if it is initially estimated larger.

Figure 4.3 shows examples of generated mixture models, sampled data and fitted models. The examples have been chosen to convey a representative impression of the variety of generated models, from well-separated components to close-to-uniform distributions in which the components are difficult to separate. Overall, fitting works well (better for $n = 1000$ than for $n = 200$), but our formal evaluation concerns that we can infer the methylation state.

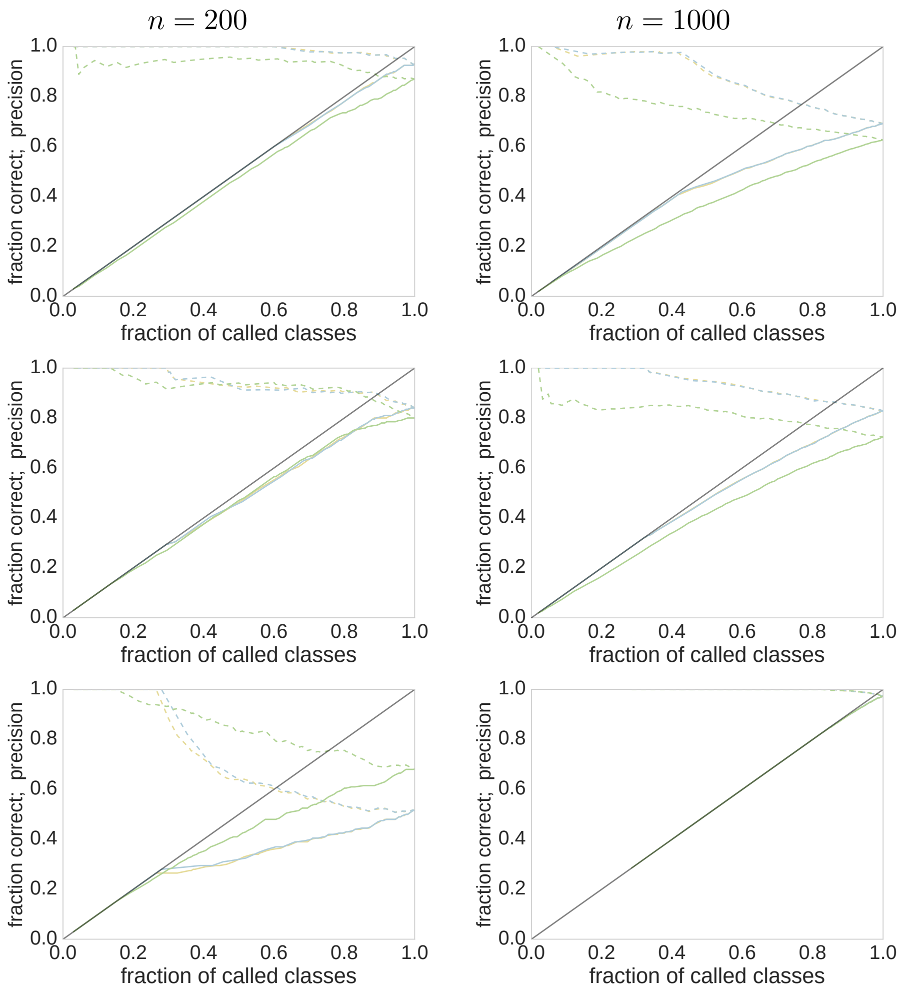


FIGURE 4.4: Performance of several classification rules. Fraction of called classes N/n (i.e., data points for which a decision was made) is shown on the x-axis against the fraction of correct classes C/n (solid lines) and against the precision C/N (dashed lines) on the y-axis for the three decision rules in Section 4.5 (green: fixed intervals; blue: highest weight with weight threshold; yellow (almost perfectly overlapping with blue): highest weight with gap threshold). The datasets are in the same layout as in Fig. 4.3.

4.5.2 Evaluation of Class Assignment Rules

Given samples $(x_1, \ldots, x_n)$ and information about which component J_i generated which observation x_i, we evaluate different procedures:

1. Fixed intervals with a slack parameter $0 \leq s \leq 0.25$: Point x is assigned to the leftmost component if $x \in [0, 0.25 - s]$, to the middle component if $x \in]0.25 + s, 0.75 - s]$ and to the right component if $x \in]0.75 + s, 1]$. The remaining points are left unassigned. For each value of s, we obtain the number of assigned points $N(s)$ and the number of correctly assigned points $C(s) \leq N(s)$. We plot the fraction of correct points $C(s)/n$ and the precision $C(s)/N(s)$ against the fraction of assigned points $N(s)/n$ for different $s \geq 0$.

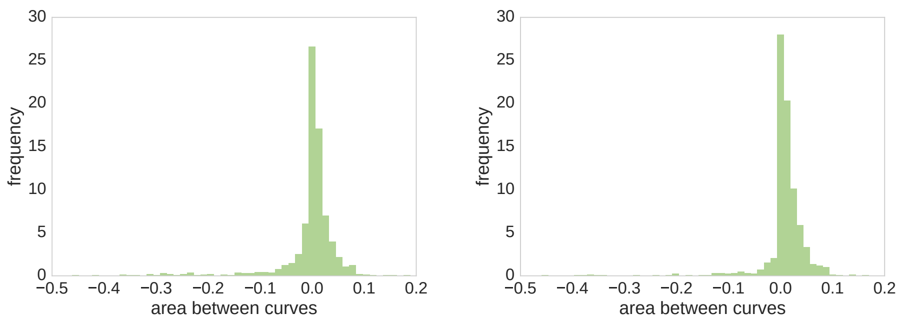


FIGURE 4.5: Signed areas between the blue curve and the green curve as in Fig. 4.4 for all 1000 simulated mixtures in dataset 1 (left; 200 samples each) and in dataset 2 (right; 1000 samples each).

2. Choosing the component with the largest responsibility weight, ignoring points when the weight is low: Point x_i is assigned to component j^* with maximal responsibility $W_i^* = W_{ij^*}$, unless $W_{ij^*} < t$ for a given threshold $0 \leq t \leq 1$, in which case it is left unassigned. We examine the resulting numbers $C(t)$ and $N(t)$ as for the previous procedure.

3. Choosing the component with the largest responsibility weight, ignoring points when the distance to the second largest weight is low: as before, but we leave points x_i unassigned if they satisfy $W_i^* - W_i^{(2)} < t$.

4. Repeating 2. and 3. with the EM algorithm instead of our algorithm would be interesting, but for all reasonable choices of ε (recall that we have to replace $x_i = 0$ by ε and $x_i = 1$ by $1 - \varepsilon$ for EM to have a well-defined log-likelihood function), we could not get the implementation in `betareg` to converge; it exited with the message "no convergence to a suitable mixture".

Figure 4.4 shows examples (the same as in Fig. 4.3) of the performance of each rule (rule 1: green; rule 2: blue; rule 3: yellow) in terms of N/n against C/n (fraction correct: solid) and C/N (precision: dashed). If a blue or yellow curve is predominantly above the corresponding green curve, using beta mixture modeling is advantageous for this dataset. Mixture modeling *fails* in particular for the example in the upper right panel. Considering the corresponding data in Fig 4.3, the distribution is close to uniform except at the extremes, and indeed this is the prototypical case where beta mixtures do more harm than they help.

We are interested in the average performance over the simulated 1000 mixtures in record 1 ($n = 200$) and record 2 ($n = 1000$). Since the yellow and blue curves do not differ significantly, we have calculated the (signed) area between the blue and green curves in Fig. 4.4 for each of the 1000 mixtures. Positive values indicate an advantage of the blue curve (classification by mixture modeling). For data set 1 we get a positive sign in 654/1000 cases (+), a negative sign in 337/1000 cases (-) and absolute differences of at most 10^{-6} in 9/1000 cases (0). For record 2, the numbers are 810/1000 (+), 186/1000 (-), and 4/1000 (0). Figure 4.5 shows histograms of the sizes of the area between the curves. While there are more cases with advantages for mixed modeling, the averages (-0.0046 for dataset 1; $+0.0073$ for dataset 2) do not reflect this due to a small number of strong outliers on the negative side. Without analyzing each instance separately, we identified the primary cause of this behavior as almost uniformly distributed data, similar to the example in the upper right panel in the figures 4.3 and 4.4 for which appropriate (but incorrect) parameters are found. A single beta distribution of $\alpha < 0$ and $\beta < 0$ would correspond reasonably well to this data, and accordingly a three-component model is not readily identifiable. Of course, such a situation can be diagnosed by calculating the distance between the sample and the even distribution and using fixed thresholds if necessary.

4.5.3 Simulation and fitting for estimating the number of components

To evaluate the component estimation algorithm, we simulate datasets with one to five components, each with $n = 1000$ samples. We simulate two different types of data sets, both using the π mixture coefficient selection method as described above.

Independent simulation

For the first kind of data, we choose components independently from each other. This frequently leads to datasets that can be effectively described by fewer components than the number used to generate the dataset. Let E be a standard exponentially distributed random variable with density function $f(x) = e^{-x}$. The parameters are chosen for each component j independently by choosing $\alpha = E_{j,1}$ and $\beta = 1 - E_{j,2}$ from independent exponentials. (If $\beta < 0$, we re-draw.)

Realistic simulation

We simulate more realistic and separable data by a second approach. The intention is to generate mixtures whose components are approximately equally distributed on the unit interval, such that each component slightly overlaps with its neighbors.

To generate a set of data points, we pick an interval $I = [E_1, 1 - E_2]$ with exponentially distributed borders. (If $1 - E_2 < E_1$, or if the interval is too small to admit c components with sufficient distance from each other, we re-draw.) For each component j, we uniformly choose a point $\mu_j \in I$. We repeat this step if the distance between any two μ values is smaller than 0.2. Sort the values such that $E_1 < \mu_1 < \cdots < \mu_c < 1 - E_2$. Let $d_j := \min[\{|\mu_i - \mu_j| : i \neq j\} \cup \{E_1, 1 - E_2\}]$. Then we set $\sigma_j = 1/4d_j$. Now μ and σ serve as mean and standard deviation for each component to generate its parameters α_j and β_j by Eq. (4.1).

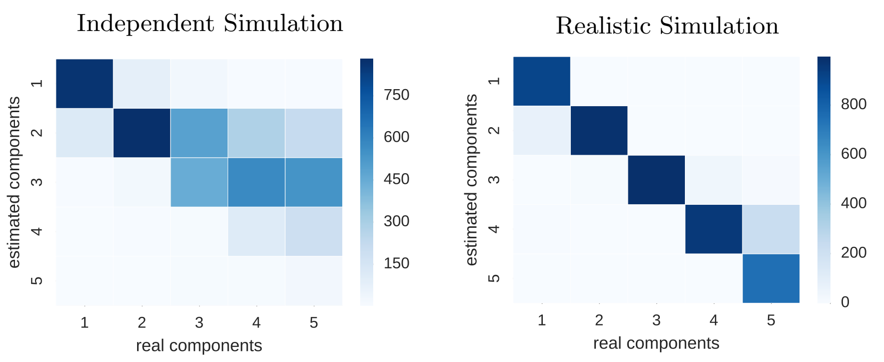


FIGURE 4.6: Comparison of the real number of components (x-axis) and the estimated number of components (y-axis) by our algorithm. Simulations consisted of 1000 datasets with 1000 data points each. Each column of each matrix sums to 1000; row sums are variable.

4.5.4 Evaluation of component estimation

We estimate the number of components as described above with a threshold d_{KS} corresponding to a p-value of ≥ 0.5 of the corresponding Kolmogorov-Smirnov test (as the fit gets better with more components, the p-value increases). The choice of 0.5 as the p-value threshold is somewhat arbitrary; it was chosen because it shows that there is *clearly no* significant deviation between the matched mix and the empirical CDF from the data; see below for the influence of this choice). We compare the actual simulated number of components with the estimated number of 1000 datasets at 1000 points each generated by (a) independent simulation and (b) realistic simulation. Figure 4.6 shows the resulting confusion matrix. Perfect estimation manifest as sharp diagonal, but we observe an underestimation of the number of components on the independently generated data, especially with higher component numbers. This is to be expected because the components of the independent simulation often overlap and lead to relatively flat and hardly separable mixture densities. For the data from the realistic simulations, we see a strong diagonal and our algorithm rarely overestimates or underestimates the number of components when the components are separable. For both data sets, our method rarely overestimates the number of components.

Choice of p-Value Threshold

In principle, we can argue for any *not significant* p-value threshold. Choosing a low threshold would result in mixtures with fewer components, which would increase under- but decrease overestimations. Choosing a high threshold would do the opposite. By systematically varying the threshold, we can check whether there is an optimal threshold, which maximizes the number of correct component estimates. Figure 4.7 shows the proportion of under- and over-estimates for both data sets (I: independent, blue; R: realistic, green), and the total error rate (sum of under- and over-estimates) for different p-thresholds. We see that the error rate in the independent model (I) is generally higher because we systematically

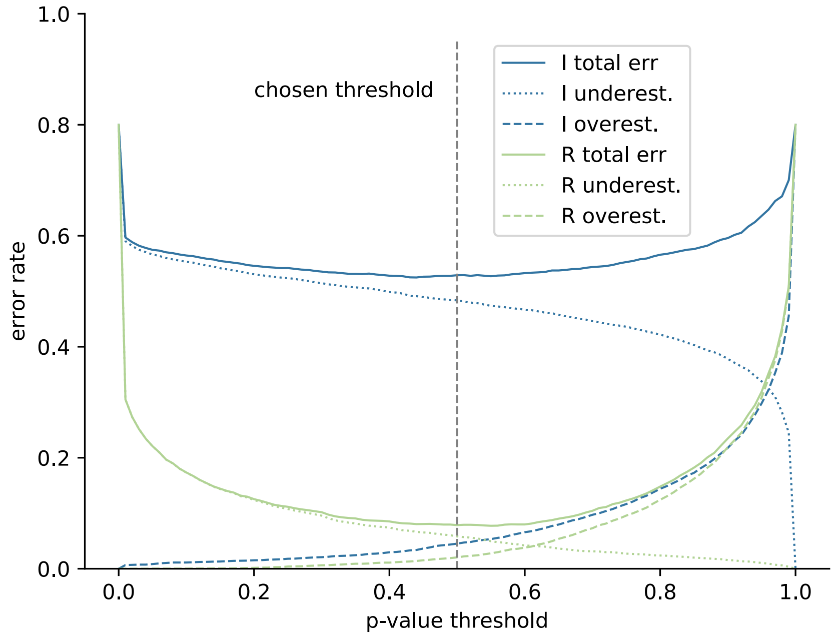


FIGURE 4.7: Fraction of under- and overestimations and total error rate (their sum) for datasets "independent" (I; blue) and "realistic" (R; green) for varying p-value threshold of the Kolmogorov-Smirnov stopping criterion when choosing the number of mixture components.

underestimate the true number of components (see above); this applies to any reasonable threshold of ≤ 0.9. We also see that both total error curves have a flat valley between 0.4 and 0.6 (or even 0.2 and 0.8), so the choice of a threshold in this range is close to the optimum; we chose 0.5 because it is a *smallest complex* in the sense of Occam's razor.

4.6 Summary

In this chapter, we present an alternative to the EM algorithm for parameter estimation of mixture models at given data points $\in [0, 1]$. We developed a hybrid approach between maximum likelihood estimation and the method of moments; it follows the iterative framework of the EM algorithm. While existing tools (with EM) have problems dealing with values of 0 and 1, our method remain robust. A comparison of our algorithm with the EM algorithm (from the `betareg` package) failed because the EM algorithm was not convergent and ended with errors. Although the method was initially developed to describe distributions of methylation levels where the values 0 and 1 are prevalent, it can be used to describe a whole family of distribution problems with data $\in [0, 1]$ from multiple components. Our approach is computationally simpler and faster than numerical ML estimation in beta distributions. Data and Python code can be obtained from the Bitbucket repository [2].

[2] https://bitbucket.org/genomeinformatics/betamix

Chapter 5

Workflows for Methylation and Exome Analysis

The previous chapters deal with methods and solutions for specific problems in the field of methylome analysis. This chapter is more descriptive and explicitly lists all steps from the raw data to the extraction of all critical information such as methylation values. Unlike most downstream analyses, this process requires no custom calculations and can be fully automated, which we implement in the workflow Methylome (*Methylome Sequencing Tooling* - `mosquito`).

Additionally, we explain *A Pipeline for Exomes* (`ape`), which can process exom data analogously to `mosquito`) and then provides the exom analysis system described in Chapter 6.

Both systems are optimized for performance, easy configuration, scalability and reproducibility. The latter is critical for performing scientific analysis, especially when multiple samples and steps are involved. We describe each pipeline step including its implementations and configurations and start with `ape` because of its easier structure.

5.1 Snakemake

Both workflows are based on the workflow managment system `snakemake`. (Köster and Rahmann, 2012). It allows the simultaneous execution of independent parts, the resumption of the workflow in case of errors, supports cluster systems and allows a simple workflow extension.

`Snakemake` workflows are composed of functional rules, each of which consists of several parameters. The most important of these are:

- input – input file(s) required to execute the rule
- ouput – output file(s) created by this rule
- shell – bash command that is executed if the rule is invoked
- threads – number of threads the bash command is using on execution
- params – a set of parameters that may be used in shell

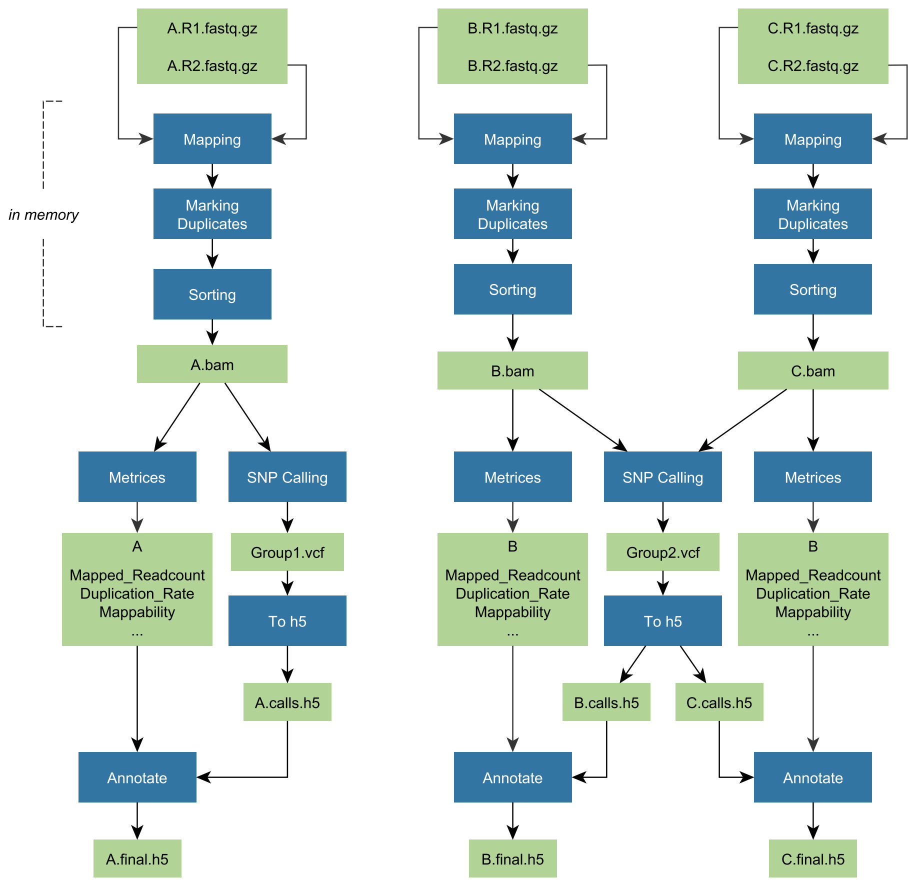


FIGURE 5.1: `Snakemake` Dependency diagram of `ape`, configured for three samples. Each command (blue) creates one or more files (green). Sample A (left) is configured for the single sample call, while B and C are processed via group calling.

When invoked, `snakemake` creates a dependency graph based on these input and output definitions and specifies the order of execution. Independent rules are executed in parallel, taking into account the maximum available CPU cores. Within a shell command, parameters are accessible via *{input}*, *{output}* and *{threads}*, and sub-elements, such as a particular input file, are addressed by the dot notation and a defined name for that element, e.g. *{input.mybamfile}*. We reduce all rule definitions in this chapter to their shell command and enter an output, as these are the essential parts to describe them.

5.2 Exome Pipeline

Exome sequencing is applied to identify SNPs and Indel on protein coding regions (see Section 1.2). We have developed our workflow *A Pipeline for Exomes* (`ape`) to process raw data from exomes and to retrieve different information as well as different statistical indicators. It performs the following steps shown in Figure 5.1.

Exome Mapping

NGS data is generated by shotgun sequencing, which produces millions of short DNA pieces. For most downstream analyses, reference aligned reads are necessary (see Chapter 1 for detailed information) which is accomplished by a so-called *read-alignment*. This process searches for the original position of all read sequences relative to the given species reference genome and determines a CIGAR string describing the differences between the read and reference sequences. We choose the widely used read mapper `BWA-MEM`(Li, 2013) for this first action of `Ape` which is to align all raw sequences. It uses a technique called *split read mapping*, which allows mapping not only the entire read sequence but also its substrings. This allows the automatic trimming of previously attached adapter sequences, which would otherwise be necessary manually. The following (partial) command is executed:

```
mkfifo {output.tmp_r1}
mkfifo {output.tmp_r2}
cat {input.fastq_r1} > {output.tmp_r1} &
cat {input.fastq_r2} > {output.tmp_r2} &
bwa mem -R {params.RG} -M -t {threads} {input.ref} \
{output.tmp_r1} {output.tmp_r2} |
```

Raw data is often divided into several compressed blocks (chunks) with a fixed number of reads. One way to deal with this is to merge the chunks into a single file and use this file as input for the read aligner. This is easily possible because a binary concatenation of gzip-compressed files is also a valid gzip file. `Ape` is designed to save as much computing time as possible. However, to save time and computing resources by reading and writing files, we avoid this step of merging. Instead, `ape` creates two named pipes by calling *mkfifo* for forward and backward reads. These are identified as files but behave like piped streams. Exactly one process can read the pipe, while precisely one process writes to this file, and the data is streamed from writer to reader process without caching. `Ape` passes the concatenated data of the strand-specific raw files to the named pipes that serve as input for `BWA-MEM` (concatenation of gz files creates valid gz files). Thus the raw data are indirectly merged by the mapping process itself.

Like most mappers, `BWA-MEM` outputs its alignments in *sam* format. Instead of converting *sam* to *bam* first, we decided to take the stream directly to the next step of *Marking Duplicates* (described below) to avoid (again) unnecessary caching on the hard disk. Note: The command is not yet finished at this point (pipe symbol at the end). To improve readability, we describe it in several parts.

Marking Duplicates

Duplicated DNA fragments produced by several PCR cycles (see Chapter 2) can lead to numerous sequenced copies of the same original reads. As usual in pre-processing, `ape` identifies and marks duplicates as such in order to avoid distortions in further downstream analyses. We have decided to use *samblaster* (Faust and Hall, 2014) for this operation. It uses a hash table to store read pairs it has already seen and can mark duplicates in a single *piped* iteration. It requires Read-id grouped (*sam* formatted) input, where paired reads are stored adjacently. This format and output structure is standard for most mappers, including BWA-MEM. A disadvantage of this approach is that the first read pair found is identified as the original read pair, which is not necessarily of the highest quality. We ignore this because current exome sequencing generates reads whose duplicates are usually of the same quality, often 60, the highest possible value. Accordingly, the command is:

```
 samblaster -M | \
sambamba view -t {threads} -S -f bam -h /dev/stdin |
```

The *samblaster* output with marked duplicates is then transformed from *sam* (text) to *bam* (binary) using *sambamba view*. Note that up to this point, all calculations are done in memory and no files were written.

Sorting

Raw data are unsorted and so are the generated *bam* files. For downstream analyses random access in constant time is desirable. An index can achieve this without having to load the large alignment files entirely into memory. Indexing the *bam* file requires the reads to be sorted by their positions, e.g. with the sambamba's *sort* operation:

```
sambamba sort /dev/stdin -m 20GB -t {threads} -o {output}
```

With a permitted total memory of 20 GB, most Exome samples can be sorted in memory. The generated sorted *bam* files contain the reads and are finally stored on the hard disk. This means that we can perform all the steps from raw reads to sorted alignments in memory without accessing the hard disk, which is usually the bottleneck.

Indexing

As an additional part of data preparation, *bam* files are indexed for efficient direct access in the form of an R-tree (Guttman, 1984). This *bam.bai* file allows direct access to any reads of a certain genomic region (chrome:start-stop) in the corresponding *bam*. (Li et al., 2009). The index is created by

```
sambamba index -t {threads} {input}
```

Providing an index, for example, allows visualization tools such as IGV (Robinson et al., 2011) to display reads of a particular region of interest without having to read the entire file first.

Variant Calling

Several different *variant callers* exist to detect variants and their genotypes from a given read alignment. For performance reasons we use the haplotype-based variant caller *freebayes* (Garrison and Marth, 2012) instead of the more commonly used GATK (McKenna et al., 2010) by

```
 './scripts/freebayes-parallel \
< (fasta_generate_regions.py {ref}.fai 10000000) \
 {threads} -u -f {ref} {input} --min-alternate-fraction 0.10 | \
 vcffilter -f "QUAL > 10" > {output}'
```

For each position with at least one or more overlapping reads, the call determines possible differences to the reference sequence and identifies its true positive probabilities. A typical study would be the identification of interesting variants by filtering calls of several samples against each other (case against control). For this purpose, we keep variants if they occur in at least 10% of all reads at this position and have a Phred-scaled quality score > 10. These tolerant parameters allow us to detect doubtful variants in healthy samples, which are then used as filters for other variants in `Eagle`. (see Section 6.4). To further improve sensitivity in low coverage areas, we use user-defined calling groups containing multiple samples, where multiple samples are called together. This approach has proven beneficial (significantly improved call quality) (Roach et al., 2010) because multi-sample variant calls can use additional information from multiple samples at a single location. Therefore, the input consists of several *bam* files, one for each sample of the corresponding group. Group calling is particularly crucial for tumor analysis, where the variant set of tumor and blood samples have a high overlap.

`Freebayes` is able to recognize SNPs and short indels (see Section 5.2), but longer structural variants (SVs) are overlooked if they exceed a certain length. Therefore `ape` also applies `Delly2` (Rausch et al., 2012):

```
export OMP_NUM_THREADS={Threads}
delly_parallel_0.7.1 -g {ref} {bam} {bam} -t {type} -o {output}
```

`Delly2` can call three different types of SVs: Deletions, insertions and inversions. Since it can only call one type of structure variant at a time, the command is executed multiple times in parallel for each sample.

Both tools output variants in standard *vcf* format (Danecek et al., 2011) with one variant per line listing the genomic position, reference, and alternative bases and counts, as well as quality values and genotype. *Vcf* files consisting of *multiple* Incoming samples generated contain genotype and count information for each sample.

Annotation of Variants

In most cases, a list of variants with basic information is not sufficient to identify interesting disease-causing genetic differences. For this reason, `ape` *SnpEff* (Cingolani et al., 2012b) to add multiple annotations to each variant. The most important annotations are: The *effect* – A sequence ontology term that describes a relationship to a particular gene (e.g., 3'UTR or intronic[1]) – and affected *genes* and *transcripts*, as well as their *putative effect* that describe the effects on protein structure.

[1]For a complete list of supported annotation terms, see `http://snpeff.sourceforge.net/VCFannotationformat_v1.0.pdf`, page 6.

Conversion

Eagle (see Chapter 6) requires data as *eagle-data-files* (see Chapter 6.3). A script provided by `Eagle` generates these files from `SnpEff` annotated *VCFs*, which is invoked by `ape` for each sample per default.

5.3 WGBS Pipeline

WGBS is used to retrieve epigenetic information in the form of CpG methylation levels. While the DNA sequence is the same in almost all (non-tumor) cells of an organism, the levels of methylation are cell- and age-specific and may change due to external influences. In most studies dealing with methylation, the information of interest is deferentially methylated regions (DMRs - see Chapter 3.3). These are not restricted to specific genomic sites such as genes. Thus, bisulfite sequencing is applied to the entire genome (WBGS), which produces significantly more data than exome sequencing (WES). This is why the requirements for a WGBS pipeline differ from those of an exome pipeline. While a sequenced WES sample consists of approximately 30 million reads, this number can increase to more than one billion for a single WBGS sample. Unlike WES, a WGBS experiment typically consists of fewer larger samples but uses multiple traces for a single sample with average coverage >20. Sequencing protocols may also vary between tracks, and data must be unified for downstream analysis. For these reasons, we divide our WGBS pipeline (*mosquito*) into a first part where individual traces are processed and a second part where data from all traces are aggregated. The first part consists of the following steps.

WGBS Mapping

The mapping process is similar to that for exonic data, except for the specialized mapper, which must be designed to map bisulfite-treated sequences, i.e. a three-letter aligner (see Section 3.1.3). We use *BWA-Meth* (Pedersen et al., 2014) to do this (with default parameters) by calling

```
bwameth.py -t {threads} -r {params.rg} \
--reference {input.ref} {input.r1} {input.r2} \
| samtools view -bS -@ {threads} |
```

`BWA-meth` applies the transformation described in Section 3.1 and uses `BWA-mem` to align the reads. It converts read sequences by replacing C → T and G → A, uses `BWA-mem` as the backend to align the reads, and then restores the original sequences. Due to the split read mapping technique, `mosquito` does not require any configuration of adapter sequences, and adapter trimming is done implicitly during the mapping process. The number of threads (currently 20) should be adjusted according to available system resources.

Sorting

Sorting is applied for each alignment by

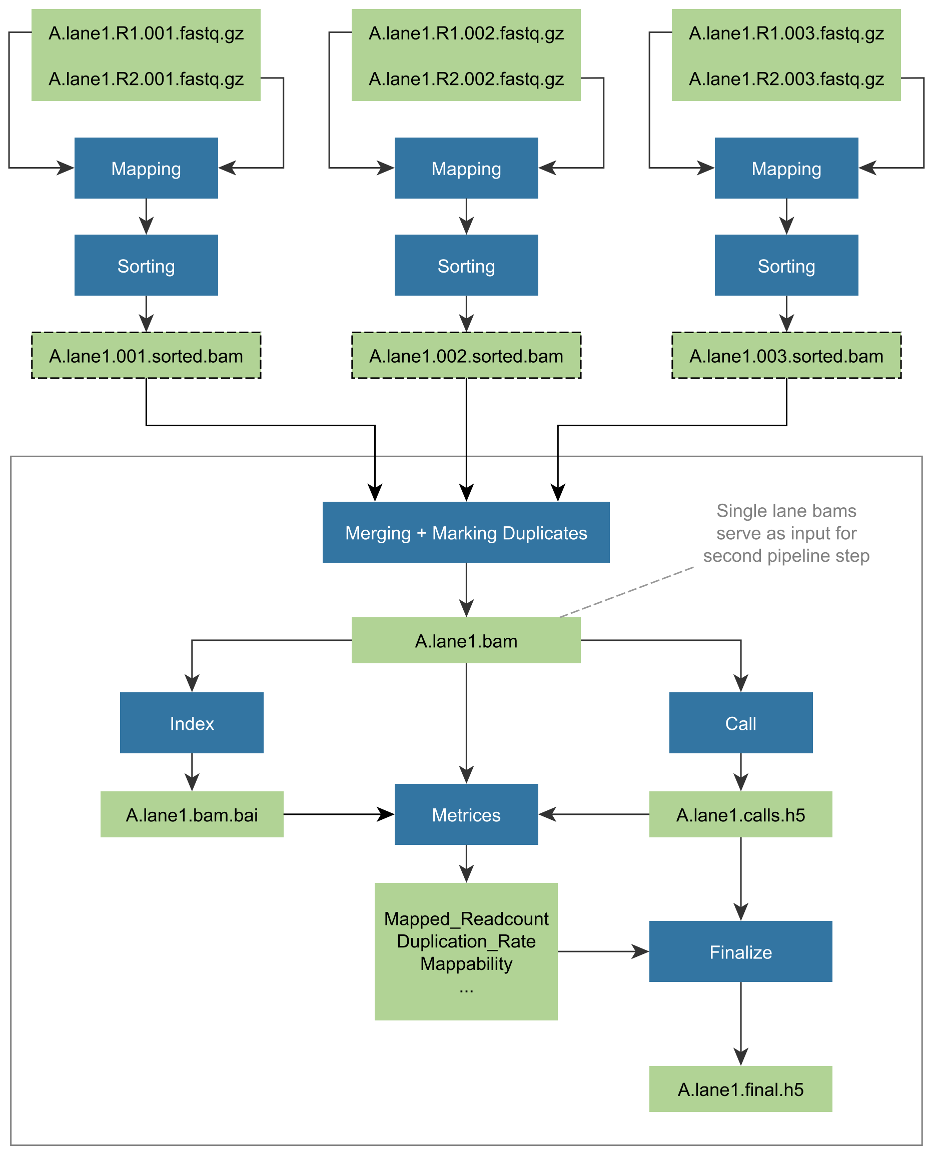


FIGURE 5.2: First part of *mosquito* exemplary for a lane *lane1* of sample *A*, divided by the sequencing process into three parts (001 - 003) with produced output files (green boxes) and executed steps (blue boxes). Temporary files (dashed black border) get deleted after execution. The upper part illustrates the separate mapping and sorting steps for the three individual paired-end reads, which are then merged into a single *bam* file. In addition to serving as input for the second part of the pipeline (Figure 5.3), these *bam* files are used to call single-track methylation levels and derive various metrics. To simplify the diagram, we have merged individual steps into a single node. Finally, metrics and calls are combined into a single h5 file. Nodes within the lower square are reused in the second part of the pipeline.

```
sambamba sort \
--tmpdir tmp /dev/stdin -m 100GB -t {threads} -o {output}
```

with 50gb maximum memory usage and 20 threads by default. Note that the alignment is directly piped to save unnecessary hard disk storing (similar to `ape`).

Merging and Marking Duplicates

Here, too, raw data can be divided into multiple chunks of several million reads each. Due to larger files, we decided to map and sort each chunk individually instead of merging them first. Sorted *bam*files are merged at the end of this process by `sambamba`, which allows simultaneous merging and marking of duplicates in a single command

```
sambamba markdup -t {threads} --sort-buffer-size 131072 \
--overflow-list-size 2000000 {input} {output.bam}
```

where input consists of all "parts" of one lane. For the large number of reads, we increased the sort buffer size to 128 GB and the overflow list size to 2000000. This reduces the number of temporary simulations accessed files, which might otherwise reach the operating system limit.

The requirement of merging chunks may be eliminated by making use of the fact that `BWA-Meth` can handle multiple input *fasta* files. Because the applied duplication mark algorithm of `sambamba` requires two iterations over *bam* files stored on disk and allows marking and merging in a single step. This does not improve *mosquito*'s performance. Replacing `sambamba` with `samblaster` would allow for piping of mapping, duplication marking and sorting without intermediate temporary storage (like in `ape`), but `mosquito` requires the second merging and mark duplication step (second part of the pipeline), where applying `samblaster` is not possible. For consistency reasons we choose `sambamba` over `samblaster` to perform both duplication mark processes equally.

Indexing

The process of indexing *bam* files is equal to the indexing process of `ape`.

```
sambamba index -t {threads} {input}
```

In principle this step is not necessary, since `sambamba merge` produces indices as a by-product. It applies in cases where the input is a *sorted* bam file but not indexed, yet. In all other cases, `sambamba` merge creates the index.

Methylation Calling

In order to obtain methylation levels for each CpG, we apply `Camel call` (Section 3.2):

```
camel call {params.nome} {input.bam} {input.index} {output}
```

The previously merged single-lane *bam* serves as input to generate *camel-call-files* containing methylation levels. We use default parameters of `camel`, i.e., minimum base quality of 17 and minimum mapping quality of 30. *Camel* is also able to call NOMe-seq GpC methylation if the respective configuration flat is set, represented by the *params.nome* parameter.

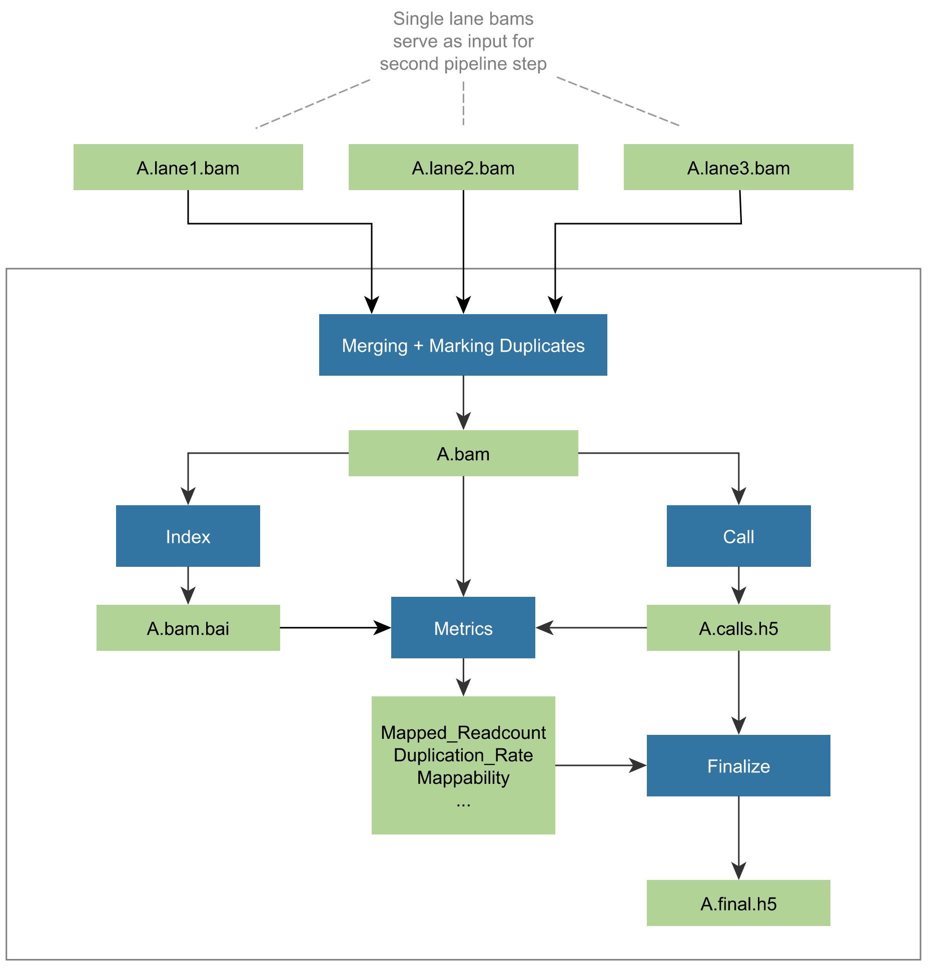


FIGURE 5.3: Second part of `mosquito` exemplary for preprocessed *bam* files of three tracks. *Call* and *Metrics* in the lower part are reused steps of the first part.

Metrics

Several metrics are calculated for each lane. For that purpose, the following counts are obtained:

- Number of all reads:
  ```
  sambamba view -c -t {threads} {input}
  ```
- Number of mapped reads:
  ```
  sambamba view -c -F "not unmapped" -t {threads} {input}
  ```
- Number of duplicated reads:
  ```
  sambamba view -c -F "duplicate" -t {threads} {input}
  ```

Lane duplication rates and mappability are also calculated based on these counts. Additionally, two methylation metrics are determined by `camel` *average* in combination with `awk`.

Mean CpG methylation, calculated by

```
camel average {input.index} {input.calls} \
| awk '{sum+=$4*$5; total+=$4} END {print sum / total}'
```

and bisulfite conversion rate by

```
camel average {input.index} {input.calls} -r L \
| tail -n +2 | awk '{print 1-$5}'
```

which describes the fraction of unmethylated cytosines that are successfully converted to thymine. This metric is calculated by adding completely unmethylated Lambda phage DNA to the experiment. The bisulfite conversion rate is $1 -$ #methylated Cs of the Lambda DNA.

Finalize finally unifies calling data and statistical data into a single *camel-data-file*.

Multi Lane Steps

Previous steps refer only to data of individual tracks. WGBS samples often require sequencing of multiple tracks to achieve desired coverage. In the end, all *bam* files are merged into a unified alignment that serves as the basis for sample calls and metrics. We process the unified *bam* by reusing the previously described steps *Merging + Mark Duplicates*, *Index*, *Call* and *Metrics*. Note that the merge of `sambamba` is performed, which also performs duplicate detection. We reuse this rule from part 1 of the workflow, but due to different read groups (commented by the host), read accesses from separate lanes are not considered potential duplicates. So this step is just a merge step. Figure 5.3 shows an example of this second part of the pipelines for three tracks (*lane1*, *lane2*, *lane3*) of a single sample *A*.

In this way, all critical information is generated for each lane and summarized for each sample. We have designed the system to allow analysis of the merged sample while still having access to all metrics and mapping data as well as the calls for each lane. This enables crucial lane-wise quality control and the detection of sequencing problems.

5.4 Summary

In this chapter we present two pipelines, `ape` for exonic data and `mosquito` for WGBS data. Both pipelines use similar steps, e.g., mapping, sorting, and indexing. Because of its multi-lane data, `mosquito` is more complex and uses a two-tier system in which data is first processed at the lane level and then uniformly as a whole. The underlying `snakemake` structure allows workflows to run on cluster systems, and both pipelines are designed to maximize CPU utilization and minimize disk space and access. Since all types of NGS data require similar necessary steps, the workflows presented here can also serve as blueprints for other pipelines such as RNA-Seq.

Chapter 6

Exome Analysis Graphical Environment

While methylome sequencing cost is still very high, whole exome sequencing (WES) has become a commonplace method to discover genes involved in, or even causing, rare heritable diseases (Maxmen, 2011; Nguyen and Charlebois, 2015) and syndromes (e.g., Wieczorek et al., 2013), with over 220 publications listed in PubMed between 2010 and early 2018. It is also a frequently used method to obtain characteristic mutations or mutational landscapes of tumors (Liu et al., 2014; Martin et al., 2013), or to characterize tumor dynamics and evolution (Masetti et al., 2016; Schramm et al., 2015). In contrast to genome-wide association studies (GWASs), direct discovery of deleterious variants is possible with WES. Compared to whole genome sequencing (WGS), WES is more cost effective because only 1–3% of the genome is captured and sequenced, i.e., regions covering at least all coding parts. The concrete definition of exome in a particular study depends on the used exome capture kit; this varies over time and from supplier to supplier.

Bioinformatics analysis of a sequenced exome consists of several steps from raw sequence data to (ideally) causative genetic variants. A typical process includes clinical data curation, sequence quality control, and trimming, read mapping and alignment, variant calling, annotation of called variants, and finally the derivation of association between variants or affected genes and disease status. Downstream, a network-based analysis of discovered genes can give further insights into disrupted pathways. For each of these steps, several methods exist, each of which has several parameters that can be adjusted, and which in turn produce additional quality metrics that can be used to guide subsequent steps in the process. For example, raw sequence reads in FASTQ files come with Phred-scale quality information, and after quality trimming and read mapping, we have mapping quality values, which quantify the uniqueness and certainty of the presumable origin of the read in the genome. Further downstream, variant calls come with several quality metrics, such as coverage and strand bias. Fortunately, best practice guidelines for exome analysis, such as the GATK Best Practices from the Broad Institute[1] have been developed from experience, which has helped to establish WES as a standard tool for genetic disease investigation. In the end, weighted, ranked and annotated results have to be interpreted by a scientist or clinician.

[1]`https://software.broadinstitute.org/gatk/best-practices/`

From a cost perspective, three factors define the costs of a WES analysis: costs for library preparation and sequencing, costs for the computational analysis (hardware, software, computing time), and personnel cost. Both sequencing costs (Muir et al., 2016) and computational costs (Schaller, 1997) have decreased exponentially; however, personnel costs remain at the same level or increase. Hence, it is desirable to automate WES analysis as much as possible and, for the final steps of reviewing and interpretation, simplify the analysis as much as possible, in order to handle the increasing amount of DNA samples assigned to each scientist or clinician for discovery or diagnosis.

We present the *Exome Analysis GraphicaL Environment* (`Eagle`), a system that allows life scientists to analyze WES data without the contiguous support of computer scientists. We designed it with the goal of minimizing the workload of everyone involved in the analysis; this concerns both the bioinformaticians who run the system and feed data into it by applying pipelines like RUbioSeq (Rubio-Camarillo et al., 2013) and the life scientists who evaluate data and results. For the latter, `Eagle` provides a web-based frontend to analyze single nucleotide variants, deletions, and insertions called from next-generation sequencing data. `Eagle` allows users to interactively explore variant calls, adjust filter parameters, view quality information, obtain aggregated summary statistics, and exchange data with other tools.

6.1 Design Overview

Exome sequencing aims to retrieve genetic sequences of protein-coding regions in order to identify disease-causing variants. A crucial step in the analysis of variant calls is filtering by attributes like quality scores, or the number of supporting samples on the one hand, and by appearance in certain groups of samples (tumor vs. normal or affected vs. not affected) on the other hand (see Chapter 6.4 for details). The former filtering is necessary to remove artifacts or to narrow the obtained results to variants called with very high confidence. The latter filtering allows scientists to investigate different questions. From our experience this filtration process is both: *manual* and *iterative*. It involves, for example, examining statistics, adjusting quality thresholds for filtering or weights for ranking variants, or manually checking interesting variants in a genome browser, such as IGV (Robinson et al., 2011), by examining the aligned reads. Systems capable of analyzing exonic data requires at least the possibility to access and treat this variant information, which can be described by:

$$\begin{tikzcd}[column sep=large] \text{Sample} \arrow[r, dash, "*"{pos=0.85}, "*"'{pos=0.15}] & \text{Variant} \arrow[r, dash, "*"{pos=0.85}, "1"'{pos=0.15}] & \text{Annotation} \end{tikzcd}$$

Each sample holds multiple variants (with different calling quality scores), and each variant may exist in one or more samples. We assume that the variant annotation is of high complexity and diversity so that each annotation exists precisely in one variant, but a variant may have multiple annotations. This is the minimum core data to deal with for a system that can

	cohorts	pairs	trios	interface	storage	install steps	install userspace
Eagle	✓	✓	✓	browser	HDF5	1: bioconda	✓
Gemini	✓	✓	✓	browser	DB:sqlite	3	✓
CanvasDB	✓	✗	✓	R	DB:MySQL	39+	✗
Var2Go	✓	✓	✗	browser	N/A	N/A	N/A
VCF-Miner	✗	✓	✓	browser	MongoDB	win installer	✓
VarSifter	✓	✓	✗	Java GUI	VCF	1	✓

TABLE 6.1: Feature comparison between `Eagle` and existing exome analysis and variant filtering tools. Column *cohorts*, *pairs* and *trios* refer to supported analysis tasks; *cohorts* means that variants of large sample sets can be compared, *pairs* refers to sets of paired samples, such as tumor/normal comparisons, and *trios* refers to family trios (mother, father, child) often examined in rare disease studies. Column *interface* refers to how the user interacts with the system; most tools nowadays use the web browser even if they run locally. Column *storage* describe which database or file format is used as backend to store and process the variants. Column *install/steps* lists the number of steps (lines) of the installation instructions. Column *install/userspace* indicates whether the software can be installed *without* administrator rights. The `EVA` tool mentioned in the text is not included in this comparison because it could not be obtained.

store and analyze exonic data. Traditional approaches of systems that filter data entries against thresholds and each other rely on multi-table databases and use backend queries for most calculations.

A unique feature of `Eagle` is that its architecture is not based on a database backend, which is in contrast to what has been considered before (Janetzki et al., 2015) and implemented in existing exome analysis systems, such as EVA (Coutant et al., 2012), GEMINI (Paila et al., 2013), CanvasDB (Ameur et al., 2014), and our own unpublished internally used `exomate` system (Martin, 2013). While this design decision may lead to a partial reimplementation of existing common database functionality, we shall argue below that it in return offers several advantages especially for the bioinformatics group or core facility that runs and maintains the system. Indeed, some existing desktop systems directly work with variant call files (VCF), such as VarSifter (Teer et al., 2012), which is typically less efficient than using an indexed and searchable database. Again other systems like VAR2GO (Granata et al., 2016) are pure cloud-based systems, where users submit their variant data online. This is often not desirable for data privacy reasons. We provide a comparison of existing systems with `Eagle` in Table 6.1.

`Eagle` is designed as a system with four layers (Files, Backend, Core, GUI; Figure 6.1). Separation into different layers allows external scripts to invoke `Eagle` functions without the graphical interface. This is useful for automated workflows in core facilities, for example. Each layer is exchangeable and only uses functions of the layer below. As `Eagle` does not use an underlying database, all data storage is file-based. The layers from top to bottom are the following.

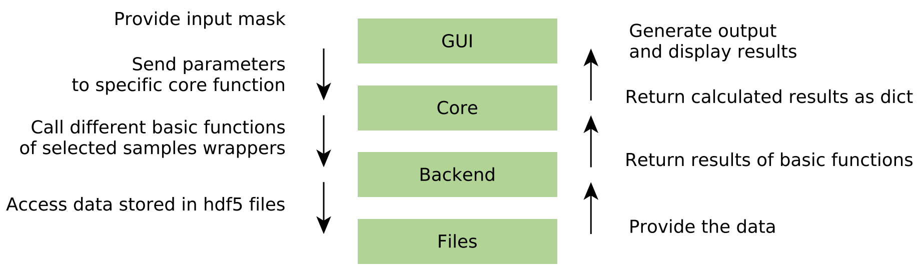

Provide input mask
Send parameters to specific core function
Call different basic functions of selected samples wrappers
Access data stored in hdf5 files
GUI
Core
Backend
Files
Generate output and display results
Return calculated results as dict
Return results of basic functions
Provide the data

FIGURE 6.1: Illustration of `Eagle`'s layers and its query, processing and answering process. Submitting a query input invokes the process. Each layer calls only functions of the layer below. Results are returned upwards and finally displayed.

GUI

`Eagle` includes a browser-based graphical user interface (GUI) to allow life scientists to perform exome analysis without the continuous support of bioinformaticians. It provides a user-friendly way to invoke the various analysis methods by the use of input masks. On submission, the query is sent to the respective function of the core layer, which in turn invokes underlying backend functions that access file data. In that way, functions are invoked through all layers, and results returned upward until a human-readable output is generated in the browser window.

Core

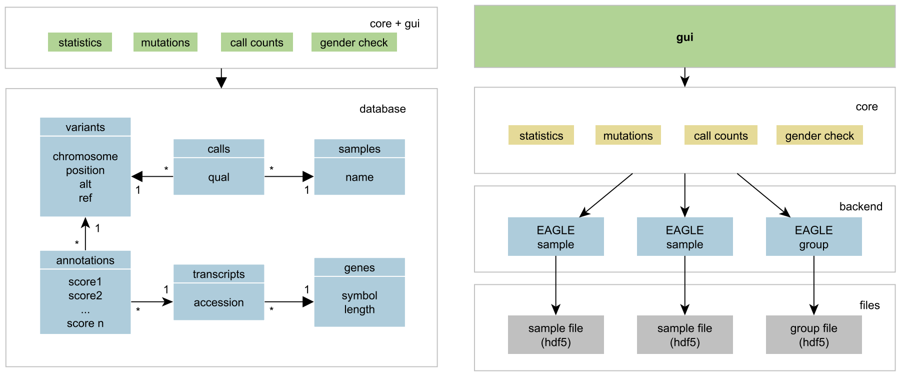
While the GUI is responsible for human-readable output, the computation of results is decoupled. All high-level functions, such as the search for causative variants or statistical calculations, are located in the core layer.

Backend

The backend is a mediator between the high-level core functions and the concrete file structure below where the data is stored. The backend layers thus provide wrappers that are used by the core layer to access the underlying file data. Currently, two wrappers exist. The *sample wrapper* enables full access to variant data for each sample. The *group wrapper* allows users to create, access, and modify group files, which are used to manage custom sample groups. The backend layer makes it possible to modify internal file structures without adapting implementations of high-level functions.

Files

`Eagle` data files are stored in structured HDF5 format (see Section (The HDF Group, 1997-2018)). Each sample file stores all called variants and statistics of exactly one sample. When working with typical workflows that produce variant call format (VCF) files, `Eagle` directly converts VCF information per sample into HDF5 files. Details of the file structure are described



(a) Database approach (b) `Eagle`

FIGURE 6.2: Comparison of system structures used by example database based approach (a) and `Eagle` (b). The database approach diagram shows GUI generation (green), responsible for data transformation and html output generation. It is separated from and built on top of a database backend, consisting of six internally connected tables (blue) and stores all relevant information. Filtering variants is performed by use of database-queries. Figure (b) shows the four layer structure of `Eagle`. From top to bottom: *GUI* (green), which generates output html and pr, *core*, which performs all calculations, *backend* (yellow), which provides basic core functionality like selecting specific data, and *files* (grey), storing the variants and statistical metrics.

in the section 6.3.

Figure 6.2 visualize `eagles`'s file based layer structure as well as an example database scheme (`exomate`).

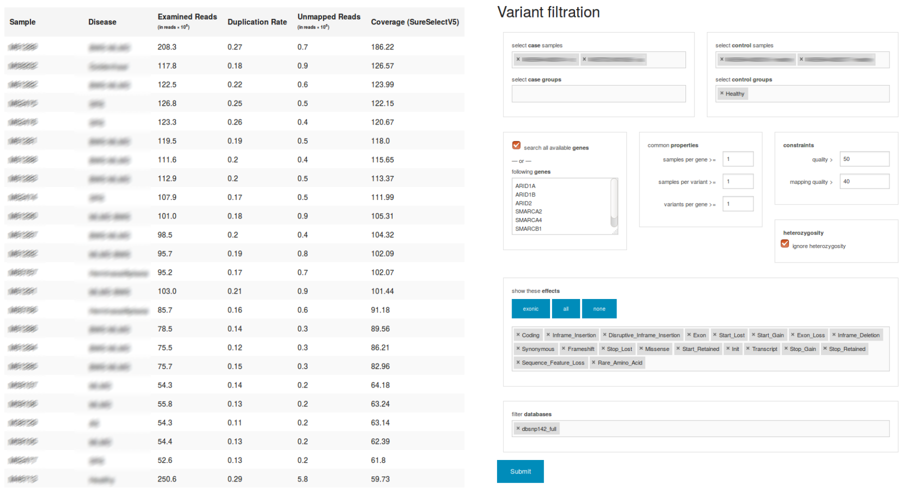
6.2 GUI

The web-based graphical user interface of `Eagle` provides three main views: the statistics view, the query view and the results view.

The *statistics view* provides summary statistics for each available sample, such as the number of mapped and unmapped reads, the estimated duplication rate of the library and its average coverage (Figure 6.3a). The provided numbers can be used for quality control, or to select suitable samples for an analysis.

The *query view* (Figure 6.3b) is the central interface of `Eagle`. It allows to select the (groups of) *case samples* of interest and also (groups of) *control samples* whose variants are likely not relevant for the question under study and which will be excluded from the reported variants. Following filtration options can be set here.

Case samples. A list of samples or a predefined group of samples defines the samples from which the variants are taken.



Sample	Disease	Examined Reads (in reads × 10^6)	Duplication Rate	Unmapped Reads (in reads × 10^6)	Coverage (SureSelectV5)
[redacted]	[redacted]	208.3	0.27	0.7	186.22
[redacted]	[redacted]	117.8	0.18	0.9	126.57
[redacted]	[redacted]	122.5	0.22	0.6	123.99
[redacted]	[redacted]	126.8	0.25	0.5	122.15
[redacted]	[redacted]	123.3	0.26	0.4	120.67
[redacted]	[redacted]	119.5	0.19	0.5	118.0
[redacted]	[redacted]	111.6	0.2	0.4	115.65
[redacted]	[redacted]	112.9	0.2	0.5	113.37
[redacted]	[redacted]	107.9	0.17	0.5	111.99
[redacted]	[redacted]	101.0	0.18	0.9	105.31
[redacted]	[redacted]	98.5	0.2	0.4	104.32
[redacted]	[redacted]	95.7	0.19	0.8	102.09
[redacted]	[redacted]	95.2	0.17	0.7	102.07
[redacted]	[redacted]	103.0	0.21	0.9	101.44
[redacted]	[redacted]	85.7	0.16	0.6	91.18
[redacted]	[redacted]	78.5	0.14	0.3	89.56
[redacted]	[redacted]	75.5	0.12	0.3	86.21
[redacted]	[redacted]	75.7	0.15	0.3	82.96
[redacted]	[redacted]	54.3	0.14	0.2	64.18
[redacted]	[redacted]	55.8	0.13	0.2	63.24
[redacted]	[redacted]	54.3	0.11	0.2	63.14
[redacted]	[redacted]	54.4	0.13	0.2	62.39
[redacted]	[redacted]	52.6	0.13	0.2	61.8
[redacted]	[redacted]	250.6	0.29	5.8	59.73

Variant filtration

select **case** samples: × [redacted] × [redacted]

select **case groups**:

select **control** samples: × [redacted] × [redacted]

select **control groups**: × Healthy

[✓] search all available **genes**

— or —

following **genes**:
ARID1A
ARID1B
ARID2
SMARCA2
SMARCA4
SMARCB1

common **properties**

samples per gene >=	1
samples per variant >=	1
variants per gene >=	1

constraints

quality >	50
mapping quality >	40

heterozygosity

[✓] ignore heterozygosity

show these **effects**

× Coding × Inframe_Insertion × Disruptive_Inframe_Insertion × Exon × Start_Lost × Start_Gain × Exon_Loss × Inframe_Deletion × Synonymous × Frameshift × Stop_Lost × Missense × Start_Retained × Init × Transcript × Stop_Gain × Stop_Retained × Sequence_Feature_Loss × Rare_Amino_Acid

filter **databases**: × dbsnp142_full

(a) Statistics view (b) Query view

FIGURE 6.3: Screenshots from two different views of the `Eagle` graphical user interface. (a) Statistics view with sample information and metrics (left to right: sample name, associated disease, number of processed reads (millions), estimated duplication rate of the sequencing library, number of unmapped reads (millions) and average coverage estimated based on the captured regions of an exome capture kit). (b) Query view: variant filtration input mask with input elements for single case and control samples, as well as case and control groups (i.e, pre-defined collections of samples), and options for selecting particular genes, various filter thresholds, effect filters, and pre-defined variant collections to be filtered out (such as dbSNP in different variations).

Control samples. A list of samples or a predefined group of control samples defines the samples from which variants are used to filter the list of case variants.

Minimum sample per variant. When searching for a specific variant, this option may help to remove variants, that are only affected by a small number of samples and are therefore uninteresting.

Minimum samples per genes Conditions of case samples are often not caused by specific variants, but by a specific affected and altered gene. Similar to the minimum sample per variant this options allows to only show variants, where minimum number of other samples have at least one variant in that specific gene. That allows to search for causing genes instead of specific variants.

Minimum variants per genes. In some cases (compound heterozygous diseases) it is of interest if a gene have more that one heterozygous variant. This option allows to search for this state.

Affected Genes. The user may limit the search for variants to specific genes, that can be defined here. Please not that gene symbols vary between different gene sets. We annotate the variants via *SnpEff* (Cingolani et al., 2012b), which internally uses the `RefSeq` database O'Leary et al., 2016 to annotate the gene symbols.

Qualities. Here the threshold for variant quality score and mapping quality is set to remove variants of low certainty.

Ignore heterozygosity. The user may set if the heterozygosity state is important for variant filtration or if variants are filtered independently from this state.

Effect. The query view can be used to exclude variants based on their effect. For example, *frameshift* or *missense* variants can be retained while *synonymous* variants are excluded (for details about effects please see Section 5.2).

dbsnp. Finally, variants from public databases like dbSNP (Sherry et al., 2001) can be used as additional control variants and *subtracted* from the results.

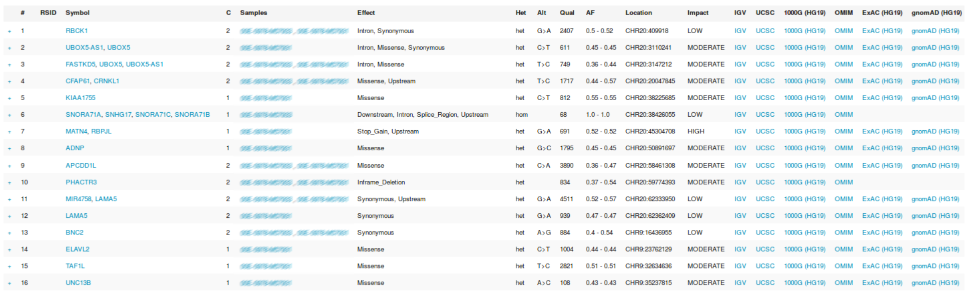
A common use case is using samples of a specific disease as case and healthy samples as control. First, the user selects case samples affected by the disease, either individually or by user predefined groups. Then he repeats this step for control samples, which are usually all available healthy samples. `Eagle` automatically provides groups for each condition, and *healthy* (consisting of all healthy samples) is preselected for convenience. For typical users, it is advised to leave most filtering options at their defaults at first, unless no or far too many results are obtained.

Finally, the *results view* is shown after `Eagle` has finished processing a query (Figure 6.4). The view shows a table with one row for each variant. The columns present various variant attributes, such as affected samples, calling quality, allele frequency, chromosomal position, and predicted effect. Shown variants are condensed to affected genes instead of transcripts, and a detailed transcript view is available via hyperlink. Variants can be viewed in external browsers like UCSC (Kent et al., 2002), 1000 Genomes (The 1000 Genomes Project Consortium, 2015), OMIM (McKusick, 2007), ExAC (Karczewski et al., 2017) and gnomeAd (Lek et al., 2016). Additionally, variants can be displayed in the Integrated Genome Viewer IGV (Robinson et al., 2011), together with the aligned reads of the selected samples.

Both the statistics view and the results view are complemented by configurable plots that enable additional quality control and ease the discovery of patterns (Figure 6.5).

6.3 File Structure

`Eagle` stores variant data in one hdf5 file (see Section 1.7) per sample, each containing one group per chromosome (Figure 6.6). Each chromosome group contains a variant table (dataset) consisting of heterogeneous information, such as the variant key (see below), and sample-specific calling quality and zygosity for each variant. The complete list of contained values is:



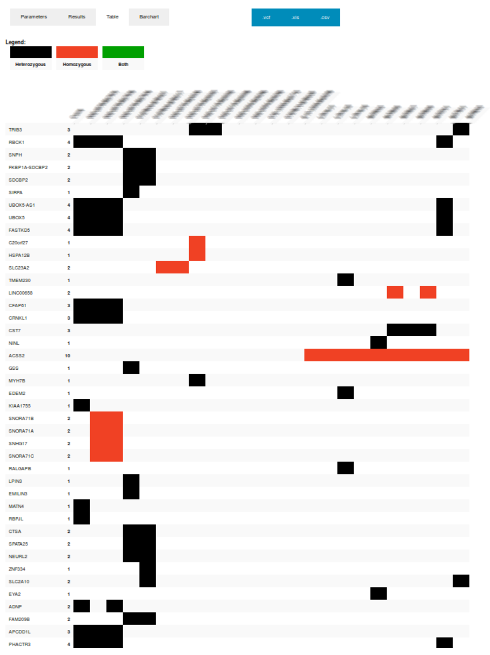
	#	RSID	Symbol	C	Samples	Effect	Het	Alt	Qual	AF	Location	Impact	IGV	UCSC	1000G (HG19)	OMIM	ExAC (HG19)	gnomAD (HG19)
+	1		RBCK1	2	[illegible]	Intron, Synonymous	het	G>A	2407	0.5 - 0.52	CHR20:409918	LOW	IGV	UCSC	1000G (HG19)	OMIM	ExAC (HG19)	gnomAD (HG19)
+	2		UBOX5-AS1, UBOX5	2	[illegible]	Intron, Missense, Synonymous	het	C>T	611	0.45 - 0.45	CHR20:3110241	MODERATE	IGV	UCSC	1000G (HG19)	OMIM	ExAC (HG19)	gnomAD (HG19)
+	3		FASTKD5, UBOX5, UBOX5-AS1	2	[illegible]	Intron, Missense	het	T>C	749	0.36 - 0.44	CHR20:3147212	MODERATE	IGV	UCSC	1000G (HG19)	OMIM	ExAC (HG19)	gnomAD (HG19)
+	4		CFAP61, CRNKL1	2	[illegible]	Missense, Upstream	het	T>C	1717	0.44 - 0.57	CHR20:20047845	MODERATE	IGV	UCSC	1000G (HG19)	OMIM	ExAC (HG19)	gnomAD (HG19)
+	5		KIAA1755	1	[illegible]	Missense	het	C>T	812	0.55 - 0.55	CHR20:38225685	MODERATE	IGV	UCSC	1000G (HG19)	OMIM	ExAC (HG19)	gnomAD (HG19)
+	6		SNORA71A, SNHG17, SNORA71C, SNORA71B	1	[illegible]	Downstream, Intron, Splice_Region, Upstream	hom		68	1.0 - 1.0	CHR20:38426055	LOW	IGV	UCSC	1000G (HG19)	OMIM		
+	7		MATN4, RBPJL	1	[illegible]	Stop_Gain, Upstream	het	G>A	691	0.52 - 0.52	CHR20:45304708	HIGH	IGV	UCSC	1000G (HG19)	OMIM	ExAC (HG19)	gnomAD (HG19)
+	8		ADNP	1	[illegible]	Missense	het	G>C	1795	0.45 - 0.45	CHR20:50891697	MODERATE	IGV	UCSC	1000G (HG19)	OMIM	ExAC (HG19)	gnomAD (HG19)
+	9		APCDD1L	2	[illegible]	Missense	het	C>A	3890	0.36 - 0.47	CHR20:58461308	MODERATE	IGV	UCSC	1000G (HG19)	OMIM	ExAC (HG19)	gnomAD (HG19)
+	10		PHACTR3	2	[illegible]	Inframe_Deletion	het		834	0.37 - 0.54	CHR20:59774393	MODERATE	IGV	UCSC	1000G (HG19)	OMIM		
+	11		MIR4758, LAMA5	2	[illegible]	Synonymous, Upstream	het	G>A	4511	0.52 - 0.57	CHR20:62333950	LOW	IGV	UCSC	1000G (HG19)	OMIM	ExAC (HG19)	gnomAD (HG19)
+	12		LAMA5	2	[illegible]	Synonymous	het	G>A	939	0.47 - 0.47	CHR20:62362409	LOW	IGV	UCSC	1000G (HG19)	OMIM	ExAC (HG19)	gnomAD (HG19)
+	13		BNC2	2	[illegible]	Synonymous	het	A>G	884	0.4 - 0.54	CHR9:16436955	LOW	IGV	UCSC	1000G (HG19)	OMIM	ExAC (HG19)	gnomAD (HG19)
+	14		ELAVL2	1	[illegible]	Missense	het	C>T	1004	0.44 - 0.44	CHR9:23762129	MODERATE	IGV	UCSC	1000G (HG19)	OMIM	ExAC (HG19)	gnomAD (HG19)
+	15		TAF1L	1	[illegible]	Missense	het	T>C	2821	0.51 - 0.51	CHR9:32634636	MODERATE	IGV	UCSC	1000G (HG19)	OMIM	ExAC (HG19)	gnomAD (HG19)
+	16		UNC13B	1	[illegible]	Missense	het	A>C	108	0.43 - 0.43	CHR9:35237815	MODERATE	IGV	UCSC	1000G (HG19)	OMIM	ExAC (HG19)	gnomAD (HG19)

FIGURE 6.4: Result view of a query. Each row corresponds to one detected variant, each column to a variant property (left to right: +: hyperlink to show detailed information on affected transcripts (the standard view shows gene symbols only); #: running number; RSID: identifier of the variant in dbSNP (if present); Symbol: list of symbols of genes affected by the variant; C: number of samples affecting the gene; Samples: list of these samples; Effect: keyword description of estimated variant effect; Het: zygosity flag (Het for heterozygous, Hom for homozygous); Alt: reference and alternative base; Qual: variant calling quality score; AF: estimated range of allele frequencies of this variant; Location: chromosome and chromosomal position; Impact: predicted impact on protein sequence; IGV: Link to open the variant in a running IGV genome browser instance; UCSC, 1000G (HG19), OMIM, ExAc (HG19), gnomeAd (HG19): external hyperlinks to genome browsers).

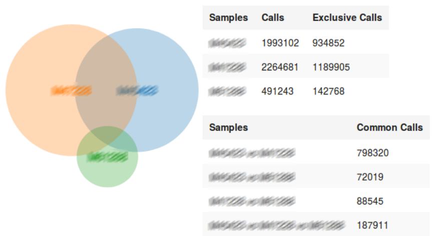
- unique 64 bit integer identification key (variant id)
- integer length of the variant ($= 1$ for SNPs, > 1 for indels)
- quality (Phred scaled false positive calling probability) of the variant
- quality-per-depth value, describing the fraction of the quality divided by the variants coverage
- the byte mean mapping quality of the variant covering reads
- 64 bit integer vector storing a bitwise-or of all single transcript effects
- 32 bit unsigned integer overall coverage and specific coverage for the alternative base(s)
- 3 byte context of this variant, character sequence of length three consisting of the variant base and flanking bases
- 32 bit rsid, the boolean *precious* flag and the boolean *common* flag from *dbsnp*, if available
- two 32 bit unsigned integers, *transcript-start* and *transcript-end*, referring to the range of transcript annotations associated with the variant.

A second dataset stores one or more variant-related transcript effects (e.g. *gene name*, *effect* or *exon number*) for each variant and the variant id for fast access. On the other hand, each variant stores its related transcripts by a range (transcript start index, transcript end index). Each sample file holds many (possibly millions of) variants, and each variant may exist in one or more samples (with different sample-specific information). The dataset contains following list of values:

- variant id



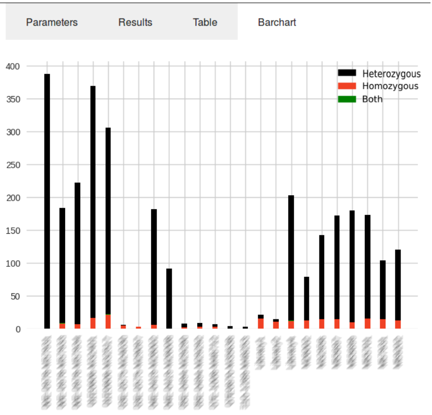
(a) Affected gene / sample matrix



Samples	Calls	Exclusive Calls
[redacted]	1993102	934852
[redacted]	2264681	1189905
[redacted]	491243	142768

Samples	Common Calls
[redacted]	798320
[redacted]	72019
[redacted]	88545
[redacted]	187911

(b) Common variants



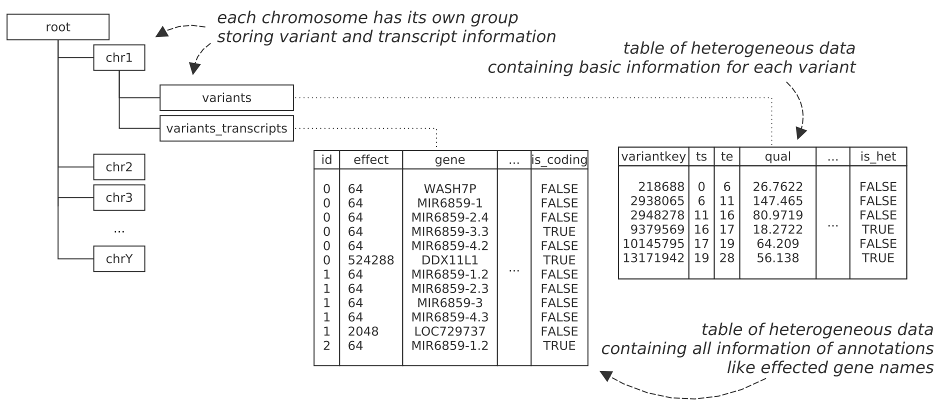
(c) Heterozygous / Homozygous statistics

FIGURE 6.5: Screenshots from various statistics. (a) Alternative view to the standard list output of variant filtration. The matrix consists of genes g (rows) and samples s (columns). Filled cells mark homozygous (black), heterozygous (red) or both types (green) of variants in pair (c, g). (b) Venn diagram illustrating variant counts and overlaps of two or three selected samples. (c) Histogram created by the row sum for each sample (column) of matrix of (a).

- 64 bit vector that stores the *effect* of the variant to the transcript
- the *gene symbol* of the affected transcript
- 32 bit unsigned integer index of the *affected exon* and *total number of exons* for the transcript
- 32 bit unsigned integer transcript id
- the *hgvsc*, which is a special representation string of a variant affecting a protein coding region
- 32 bit integer *distance to the transcript*, if the variant does not overlap with the transcript, but is in proximity
- the byte *impact*, describing the probability to change the resulting protein structure and functionality represented by the transcript
- boolean flag if the variant is inside a coding region

6.4 Variant Filtration

The central functionality of `Eagle` is the fast and interactive filtration of variants. In the following, we outline how this is achieved.


root
chr1
chr2
chr3
...
chrY
variants
variants_transcripts
each chromosome has its own group storing variant and transcript information
table of heterogeneous data containing basic information for each variant
table of heterogeneous data containing all information of annotations like effected gene names

id	effect	gene	...	is_coding
0	64	WASH7P	...	FALSE
0	64	MIR6859-1		FALSE
0	64	MIR6859-2.4		FALSE
0	64	MIR6859-3.3		TRUE
0	64	MIR6859-4.2		FALSE
0	524288	DDX11L1		TRUE
1	64	MIR6859-1.2		FALSE
1	64	MIR6859-2.3		FALSE
1	64	MIR6859-3		FALSE
1	64	MIR6859-4.3		FALSE
1	2048	LOC729737		FALSE
2	64	MIR6859-1.2		TRUE

variantkey	ts	te	qual	...	is_het
218688	0	6	26.7622	...	FALSE
2938065	6	11	147.465		FALSE
2948278	11	16	80.9719		FALSE
9379569	16	17	18.2722		TRUE
10145795	17	19	64.209		FALSE
13171942	19	28	56.138		TRUE

FIGURE 6.6: `Eagle` sample file structure. The data is organized as tree. There is one group for each chromosome. Each chromosome group contains data about variants in tabular form (numpy record arrays; see text).

6.4.1 Variant keys

To perform variant filtering, each variant must be uniquely identifiable. We conceptually define the key of a variant as a 4-tuple of chromosome, position, reference allele sequence and alternative allele sequence. However, operating on 4-tuples as keys would be computationally expensive on large numbers of variants. Therefore, we encode each variant as a 64-bit integer key (Figure 6.7) in the following way.

First, note that the chromosome can be omitted, as they are encoded in the tree structure of the `Eagle` sample files. (The processing of queries occurs chromosome-by-chromosome, and independent chromosomes may be processed in parallel, although this is currently not done). Also note that the reference allele sequence can be deduced from chromosome, position and reference allele length, so it does not need to be explicitly stored.

The least significant bit (bit 0) holds the heterozygosity state of the variant in the sample, where 0 represents an estimated homozygous variant, and 1 represents a heterozygous variant. While this flag is not part of the variant key itself, it is convenient to have it encoded in the same integer for filtering; it can always be ignored by right-shifting the keys by one bit. Bits 1–3 encode the variant type: The are four single nucleotide variant (SNP) types, one with alternative allele A, C, G , and T each. Because most stored variants are SNPs, it helps to encode them specially. The fifth type is a deletion, whose length is stored in subsequent bits. The sixth type is an insertion, where the inserted sequence needs to be uniquely identified (using subsequent bits). The remaining types are reserved for multiple nucleotide variants (which are not supported yet and ignored at the moment). Bits 4–31 encode the position of the variant. Note that 28 bits are sufficient to encode any position in the largest human chromosome. Bits 32–63 encode information about deletions and insertions. Since there may exist different deletions or insertions at the same position, each with a different length or insertion sequence, the key needs to be able to distinguish between them. In case of a deletion, the 32 bits store the length of the deleted sequence, and

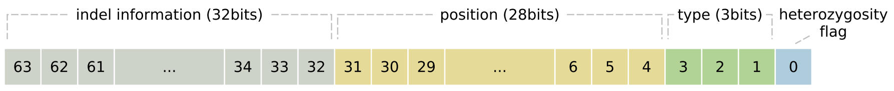


FIGURE 6.7: The 64-bit variant key. The least significant bit (bit 0; blue) stores the heterozygosity state of the variant in the sample. Bits 1–3 (green) store the variant type, bits 4–31 (yellow) store the chromosomal position, and bits 32–63 (grey) store indel information (see text).

the deleted sequence is given by the (globally known) reference genome. For insertions, a hash value of the inserted sequence is stored; to be precise, the variable length *SHAKE-128* algorithm, set to 32 bit output. (While it is possible that hashes of two distinct sequences collide, the probability is extremely low that this happens at the same position. If it does happen, the system treats two different insertion variants as the same one.)

The variant keys are computed for every variant in every sample. Variant information is sorted by the variant key in each chromosome group in each sample file. Then, variant filter operations can be performed via combinations of linear time set operations over the key vectors.

6.4.2 Standard variant filtration

Let S be the set of all samples available in an `Eagle` instance. Let $A \subseteq S$ be the set of selected case samples, $O \subseteq S$ be a disjoint set of selected control samples and let V_s be the set of all variant calls in sample $s \in S$. Filtration is carried out in three steps:

1. For each sample $a \in A$, a subset $V_a^1 \subseteq V_a$ is generated containing the set of variants in V_a that meet the selected per-sample filter criteria (i.e., quality thresholds).
2. For each sample, a subset $V_a^2 = V_a^1 \setminus \bigcup_{o \in O} V_o$ is generated by removing all variants that occur in at least one control sample. The user can decide whether whether variants of different zygosity are considered as the same or not.
3. The final set of variants $V \subseteq \bigcup_{a \in A} V_a^2$ is generated by taking the union of all variants of the selected case samples filtered for global criteria like minimum number of samples supporting the variant.

Note that in steps 1 and 2, calculations can de done independently, and hence in parallel, for each case sample $a \in A$ (this is presently not implemented).

6.4.3 Child-parents trio filtration

A special type of analysis that cannot be handled by standard filtration is the investigation of child-parent trios. This can be important to understand genetic causes of a disease the child is suffering from. If the parents are healthy, one may hypothesize that the disease is caused by recessive variants, such that symptoms do not occur as long as the affected gene has at least one working copy. Here, two cases can be distinguished.

First, both parents can exhibit the same variant, which then occurs as *homozygous* in the child. For this case, above filtration mode can be used directly.

Second, both parents can exhibit different variants affecting the same gene, which then occur as two heterozygous variants in the child. This is called *compound heterozygosity*. Here, we can filter by seeking for variants where at least one additional variant exists that targets the the same gene. Both variants must be heterozygous, one inherited from mother, the other from the father.

We now describe these cases formally. Let $c, f, m \in S$ denote the child, father and mother sample, respectively. Let $g(v)$ be the set of genes targeted by variant v. Let $\tilde{V}_s^1 \subseteq V_s^1$ be the subset of *heterozygous* variants and $\overline{V}_s^1 = V_s^1 \setminus \tilde{V}_s^1$ be the subset of *homozygous* variants from those obtained for sample s in the first of above filtration steps. The candidate set of homozygous variants for c is given by

$$\overline{V}_c^1 \cap \tilde{V}_f^1 \cap \tilde{V}_m^1,$$

the set of compound heterozygous variants by

$$\{x \in \tilde{V}_c^1 \mid \exists\, y \in \tilde{V}_c^1,\ y \neq x,\ g(x) \cap g(y) \neq \emptyset,$$
$$(x \in \tilde{V}_f^1 \wedge y \in \tilde{V}_m^1) \vee (x \in \tilde{V}_m^1 \wedge y \in \tilde{V}_f^1)\}\,.$$

6.4.4 Performance Example

As an example for a typical in-house use of `Eagle`, we provide numbers on a dataset consisting of 10 individuals (case samples) with a rare congenital defect. Here, a typical question would be which variants occur only in individuals suffering from the syndrome. To obtain these, we queried for all variants with a minimum quality of 200, a minimum average mapping quality of 40, and filtered against an in-house cohort of 259 healthy individuals (control samples). In addition, we remove all variants that are marked as being non-clinical in dbSNP.

The case samples exhibit in total 17,668,654 variant calls (avg. 1,766,865 per sample), the control samples exhibit in total 253,824,358 variant calls (avg. 1,450,424 per sample). On a harddisk-based server (no SSD usage except for the operating system), on a single core of an AMD Opteron 6276 CPU, the query is calculated in 3 minutes.

6.5 Features for Bioinformatics Teams

`Eagle` was designed not only as a user-friendly system, but also as a system that is easy to set up and maintain for bioinformatics support groups or core facilities.

Installation and setup

Installation and maintenance of `Eagle` is particularly simple, without the need to configure a database engine, or obtaining administration privileges. There are no dependencies except a number of commonly used Python packages. As `Eagle` is available as a Bioconda[2] package, it can be installed (including all dependencies) with a single command

```
conda install -c bioconda eagle
```

After that, only minimal configuration (setting paths and genome versions) is required. Detailed instructions, alternatives, and configuration examples can be found at https://bitbucket.org/christopherschroeder/eagle.

Sample handling

Because each sample is stored in a separate file in the filesystem, it is straightforward to add, remove, relabel, and backup samples. Moreover, a sample file can be easily exchanged between instances or even labs.

`Eagle` assumes that the endpoint of an exome processing workflow is one or more variant call format (VCF) file(s) that have been annotated with SnpEff (Cingolani et al., 2012a). SnpEff estimates the effect of variants and annotates whether they are contained in public databases such as dbSNP. These files must be initially converted to one `Eagle` HDF5 file per sample. During this process, the relevant fields of the VCF records are converted into the HDF5 based format. Once the files are registered in the `Eagle` configuration, the system is ready to be used for analysis of the variant calls via the GUI.

Importing new samples entails running an automated script to convert called variants in VCF format to `Eagle`'s internal HDF5 format and making the new samples known to `Eagle` by assigning a label, such as 'healthy' or a particular disease name, to each sample.

Fewer communication iterations

Variant calling workflows are typically tuned in such a way that they output a reasonable number of variants to be shown to the clinician or biologist

[2] https://bioconda.github.io

under typical sequencing conditions. However, the optimal workflow parameters may change on a case-by-case basis. For us and others, this has sometimes resulted in requests to re-run the analysis to produce a larger/smaller set of variants. The idea behind `Eagle` is that a standard workflow is run only once, but with very relaxed parameters, producing large candidate variant sets, resulting in large HDF5 files as well. From that starting point, `Eagle` can be used to dynamically adjust any filter threshold to a more stringent one and dynamically re-compute the output set. `Eagle` is able to deal with very large variant sets in reasonable amounts of time (see performance example and example use case below).

Usage as library

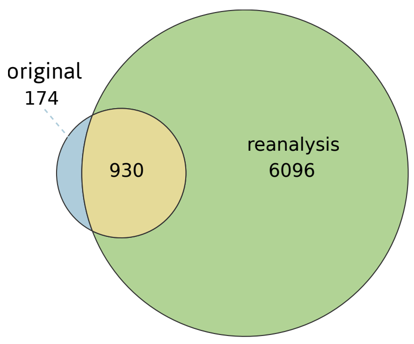
Because of its layered architecture, `Eagle` is not only a frontend but can also be used as a library. Hence, it can be accessed from analysis workflows that perform additional postprocessing or create publication quality figures in a reproducible and automated way.

6.6 Example Use Case

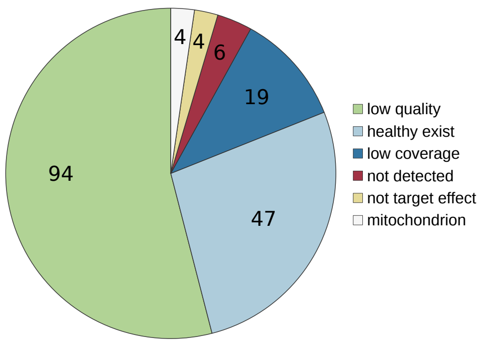
During the last years `Eagle` improved performance in handling and calculation. We show these improvements be reproducing the results of Schramm et al., 2015. In this work we identified tumor and relapse specific genetic variants using 16 neuroblastoma triples: tumor, relapse and normal tissue for 16 different donors. Calling and analysis was done by a `Exomate` (and its preprocessing pipeline) and results were filtered with a quality threshold of 200 (false positive probability $< 10^{-20}$) and a mean mapping quality threshold of 40. We reprocessed the neuroblastoma raw data using `Ape` and `Eagle` (using the same thresholds).

The first thing that we noticed is that we obtained much higher quality scores. We assume that this is due to the fact that `Ape` employs *group calling*, where where related samples are called together, leading to higher confidence. As shown in figure 6.8, we detected 7026 genetic variants in our reanalysis, while the original work describes 1104. 930 of these variants occur in both sets, 174 are exclusively represented in the original work and 6102 newly discovered variants, which appear due to the threshold being smaller relative to the overall higher quality scores. Since we investigate the difference between both systems, this intersection is not relevant to us. We had a deeper look at the 174 missing variants to identify the reasons *why* these were missing:

- We found that 92 of these variants are suppressed as their *calling quality* was below the threshold and 2 are suppressed as their *mapping quality* was below the threshold. We call these variants *low quality*. In principle these could be detected with lower thresholds.
- 47 variants were not only called in the expected samples, but also in their associated control blood samples. The pipeline of `exomate` was



(a) variant overlap of both analysis



(b) type of undetected variants

FIGURE 6.8: (a): Number of exclusive variants found in the original analysis (blue), our reanalysis (green) and common variants contained in both (yellow). (b): The 174 undetected variants classified by reason of non detection.

not able to identify these. We were not able to find any evidence for a miscalling and conclude that these are false positives of the original work.

- `Eagle` is not yet configured to show mitochondrial variants, which excludes 4 variants.
- Another 4 variants have had an effect assigned which was not selected for the analysis (intergenic and downstream). An analysis with parameters where these effects were added to the selection detects them, but also largely increase the number of additional (probably uninteresting) variants.
- Another 19 variants, all of them insertions, are not detected due to low coverage for the alternative allele of 3 or less. We cannot exclude the possibility that some of them might be undetected false negatives, especially if coverage is low and the relative alternative allele high. 4 of these 19 are likely true negatives (noise / originally miscalled), due to their coverage of about 120.
- Finally, 6 of the variants are undetected by `Ape` and `Eagle` for unknown reasons.

Of all 174 undetected variants (low quality, different target effect and mitochondrial), 102 could be detected with different parameters or the inclusion of the mitochondrial chromosome to the system. 47 of the 174 variants are assumed to be false positives in the original work and the number of missing variants lies between 6 to 25, depending on the real genotype of the low coverage variants. Figure 6.8 shows a visualization of these numbers. If we include the numbers in the overlap, we summarize that there are at most $1104 - 47 = 1057$ true positives in the original analysis and we are able to find $930 + 92 + 4 + 4 = 1030$ (97.4%) of the given variants. We conclude that we were not only able to reproduce the main results, but also improve them. Group calling and higher quality scores allow us to detect a much higher number of genetic variants and detect false positives in the original work. It would be interesting to incorporate the new variants and reanalyze the original study in cooperation with a geneticist.

6.7 Implementation

`Eagle` is implemented in Python 3, using `scipy` and `numpy` (Walt et al., 2011) for all computationally demanding tasks. In particular, no custom C extensions are required. The webserver and the graphical interface is implemented with `flask` (Ronacher, 2017) and `jinja2` (Ronacher, 2008) in combination with foundation (ZURB, Inc., 2008). Dynamic server-side graphics are generated via `matplotlib` (Hunter, 2007), client-side graphics by NVD3 (Novus Partners, 2014). We use the python provided 128 bit hash function *SHAKE-128*, which allows arbitrary output length (set to 32 bit) on the insertion sequence string to generate the insertion sequence hash. The underlying HDF5 (The HDF Group, 1997-2018) files are created, modified and accessed by h5py (Collette, 2008).

The variant filtration operations described above are implemented with appropriate `numpy` functions. For example, one step consists of generating a common set of control variants by first concatenating variant keys from all control samples (`numpy.concatenate`) and merging them into a set of unique keys (`numpy.unique`). For sorted arrays containing unique elements, `numpy.in1d` is used for membership testing.

6.8 Summary

In this chapter we present `Eagle`, a framework and graphical environment for interactive analysis of variant calls from exome sequencing data. The system is specifically designed for intuitive interactive exploration and quality control, reducing the otherwise required error-prone iterative communication process between life scientists and computer scientists. In contrast to other tools, data is stored in sample-wise encapsulated (HDF5) files, offering easy setup and maintenance without the requirement to setup and configure a database backend, while achieving the same runtime complexity for queries as relational databases.

Naturally, manually writing code for specific filter queries on HDF5 files instead of using an object-relational database system comes with implementation overhead. However, note that the filtering needed here is of an algebraic nature; the main operations are forming set unions and set differences of variants, while applying filtration criteria on different levels at the same time. Compared to what object-relational databases offer, this is a very narrow use case that, in our opinion, does not justify the overhead of setup and maintenance of a database system. Moreover, the extensive and fine-grained control that a low-level storage like HDF5 offers allows for additional optimization, which we strive to explore in future work. A unique feature of HDF5 is the ability to combine column-wise and row-wise storage. This enables the optimization of the table structure to different access patterns. Together with the numerous transparent compression filters offered by HDF5, we plan to leverage this to further tune the practical performance of `Eagle`. For example, for fast filtering, the `variantkey` column (see Figure 6.6) can be stored as a single continuous block which can be loaded in one operation without overhead. In contrast, when variants have been filtered, only certain rows in the rest of the `variants` table have to be accessed. Hence, it is reasonable to choose row-wise storage for everything except the `variantkey` column.

We plan to further extend `Eagle` in the future, by, e.g., allowing to persist query results and allowing to access them later both in the frontend and via the library through an accession. Moreover, we will investigate the scaling of `Eagle` to whole genome variant analysis and whole genome methylation analysis.

Chapter 7

Analysis of Interindividual Methylation Pattern

In one of our studies, we investigated the differences between monocytes and macrophages (Wallner et al., 2016) using the methods described in the previous chapters. Two male donors each provided four different cell types: Monocytes (MoCt), macrophages (MaCt), macrophages with E-LDL (MaTe) or Ox-LDL (MaTo) (two different types of lipoproteins). We have performed three analyses:

1. Monocytes vs. macrophages (MoCt vs. MaCt)
2. Macrophages vs. E-LDL macrophages (MaCt vs. MaTe)
3. E-LDL macrophages vs. Ox-LDL macrophages (MaTe vs. MaTo)

We processed all samples with `mosquito` (Chapter 5.3) and applied `bsmooth` to all three groups. While we could not detect methylation differences for MaCt vs. MaTo and MaCt vs. MaTe, we identified 114 DMRs between monocytes and macrophages (MoCt vs. MaCt).

A principal component analysis (PCA), shown in Figure 7.1 (a), of methylations levels of the $2 \cdot 4 = 8$ (donors $\cdot$ cell types) methylomes reveals not only differences between cell types, but also between both donors. The PCA shows greater differences for donors than for cell types, and we have investigated these interindividual methylation differences in a second study (Schröder et al., 2017).

This work differs from most other epigenetic research because it deals with differences between several individuals and not with differences between two groups, as well as genetic information. In this chapter, we describe the bioinformatics methods we have developed and the standard analysis steps we have applied. These can be used directly or with slight adaptation to any methylome analysis and are also applicable to normal DMRs between groups.

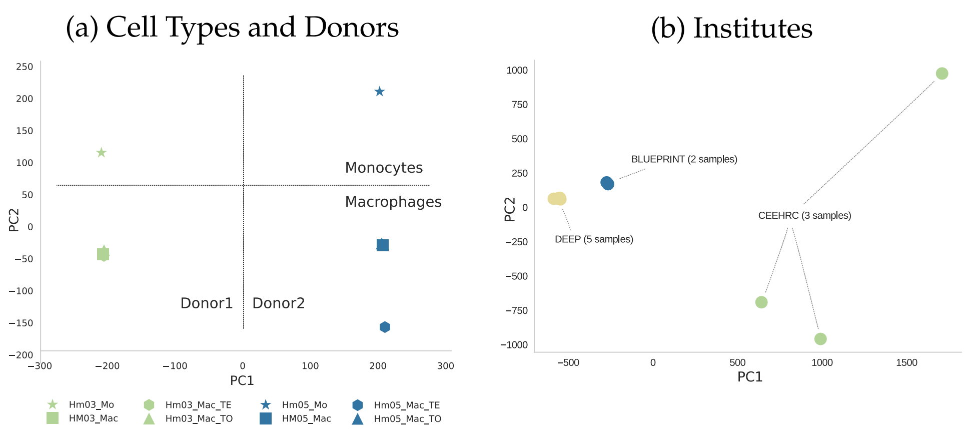


FIGURE 7.1: (a) Principal component analysis (PCA) of all methylation levels of CpGs with coverage > 10 of four different cell tissues (monocytes, native macrophages, Ox-LDL-loaded, and E-LDL-loaded macrophages symbolized by different forms) from two different donors (green and blue). The figure shows differences between monocytes and macrophages (top vs. bottom), as well as (larger) individual differences between both donors (left vs. right). b) PCA of monocyte methylomes obtained from DEEP (yellow), BLUEPRINT (blue) and CEEHRC (green). The analysis shows institute-wide clustering, probably due to different sequencing protocols and bioinformatic processing.

7.1 Detecting Interindividual DMRs

Usually, analyses of epigenetic data (such as methylomes in Wallner et al., 2016) are performed on groups of several samples. Methods for methylome comparison, including the DMR call of `Camel` (Chapter 3.3), were therefore developed for work with two groups. The identification and characterization of DMRs between several individuals without biological replicas require different approaches and methods.

7.1.1 Methylome Data

The study was performed on the original monocyte methylomes from two donors and additional monocytes from three donors, using a total of five ungrouped and independent samples. Since an increased number of samples significantly improved detection sensitivity, we considered adding methylomes produced by other institutes within the IHEC (Stunnenberg et al., 2016). We rejected this idea because a PCA of four DEEP, two BLUEPRINT and three CEEHRC methylomes retrieved from the IHEC data portal (Bujold et al., 2016) showed clear institute-specific clustering as shown in Figure 7.1. (b). This is probably due to differences in sequencing and post-processing steps, which means that mixing methylomes from different institutes can cause distortions.

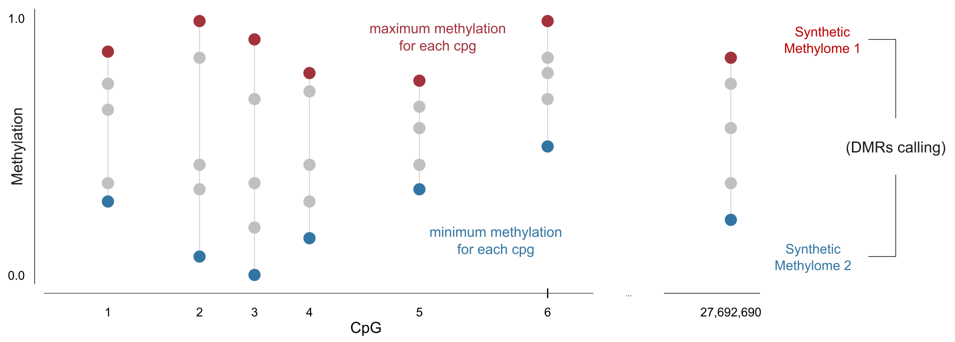


FIGURE 7.2: Creation of two synthetic (maximum and minimum) methylomes. The maximum synthetic methylome (red) consists of the maximum methylation concentrations of all samples for each CpG and the minimum values for the minimum synthetic methylomE (blue). These methylomes are used as inputs for the DMR calling.

7.1.2 Calling by Synthetic Methylomes

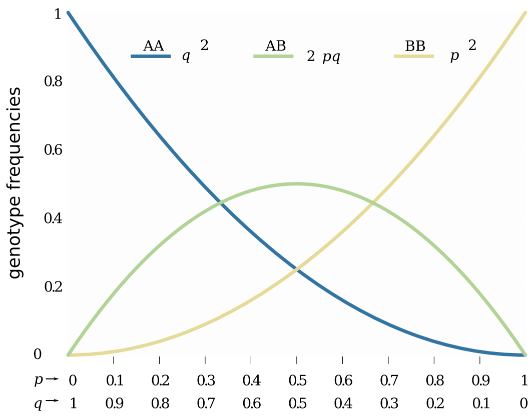
We processed the data regularly with `Mosquito` and determined the methylation level for each sample. To call DMRs between independent methylomes, we created two synthetic methylomes generated from the five samples. One consists of the maximum, the other of the minimum methylation level of each CpG, illustrated in Figure 7.2. These two synthetic methylomes then serve as two groups (of one sample each) for a DMR calling algorithm that supports groups of size one.

We applied the approach described in Chapter 3.3 and identified 157 interindividual DMRs of at least 4 CpGs and a mean methylation level difference greater than 0.8. The DMRs comprise 1165 CpGs with a total size range of 9-1,495 bp and 4-44 CpGs. Five of these DMRs have already been reported by others: Two regions (DMR87 and DMR134) overlap with the named hap-ASM-DMRs (Do et al., 2016), two DMRs contain a reported SNP mQTL (DMR25 - rs6760544), Schalkwyk et al., 2010) and DMR104 - rs11158727 (Gibbs et al., 2010)) and a DMR contains a reported ASM-SNP (DMR24 - rs1530562 (Paliwal et al., 2013)).

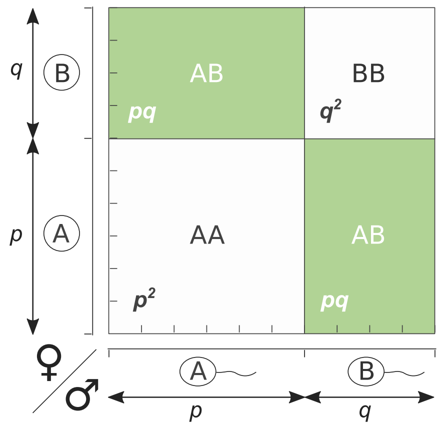
Note that the described approach of detecting DMRs with synthetic methylomes is only suitable for a small number of samples. If applied to a large number of methylomes, outliers would increase the false-positive rate of DMRs detected. This can be avoided by, for example, using the n-th lowest and highest values.

7.1.3 Statistical Significance

We have developed a method to test DMRs for their statistical significance. First, we calculate an empirical p-value by simulating 1000 sets of five samples from the null model that there is no methylation difference as follows:



(a) Genotypes distribution.



(b) Lengths of p, q correspond to allele frequencies. Area of rectangles represents genotype frequencies.

FIGURE 7.3: Visualization of the Hardy-Weinberg equilibrium. The distribution of the genotypes depending on the allele frequency p (a) and the general Punnett-Square (b) visually explaining the genotype frequency for two alleles.

Let $n_{s,c}$ be the coverage and $m_{s,c}$ the methylation count for observed sample s at CpG c. The average methylation level is given by

$$p_c = \frac{\sum_s m_{s,c}}{\sum_s n_{s,c}}$$

for each CpG c. For each of the five samples s we simulate a corresponding null sample o: We set the coverage of CpG c in sample o to $n_{o,c} = n_{s,c}$ and methylation counts $M_{o,c}$ for each c are randomly chosen with binomial probability

$$P(M_{o,c} = m) = \binom{n_{o,c}}{m} \cdot p_c^m \cdot (1 - p_c)^{n_{o,c}-m}$$

Therefore, the coverage of each CpG in each simulated null sample is equal to the coverage in corresponding observed samples, while differences in methylation are caused only by the finite sample size. By definition, there are no DMRs for the null samples and any DMR detected is a false positive. We have applied our DMR detection algorithm to the null samples. The algorithm did not detect DMRs in one of 1000 replicates, resulting in an empirical p-value < 0.001 for each DMR. This general approach to statistical significance can be applied to any DMR detection method because the null model is independent of the method.

7.1.4 Missing DMRs

Our analysis is based on only five samples, and strict settings allowed us to detect only DMRs that were homozygously methylated in at least one individual and homozygously unmethylated in at least one other individual. We can estimate the number of undiscovered DMRs in the human population that meet the same criteria. Assuming that exactly one SNP is responsible for the methylation state of a DMR and that the probability of being causal is independent of its allele frequency, we conclude that methylation states are related to genotypes of causal SNPs, and therefore epigenetic genotypes equally follow the Hardy-Weinberg equilibrium.

The Hardy Weinberg Equilibrium. In case of a single locus with two alleles A and B and allele frequencies $f(A) = p$ and $f(B) = q = 1 - p$, the expected genotype frequencies are $f(AA) = p^2$ for AA homozygotes, $f(BB) = q^2$ for BB homozygotes and $f(AB) = 2pq$ for heterozygotes. Figure 7.3 visualize the Hardy-Weinberg equilibrium.

Our approach and the chosen parameters are only able to detect DMRs if at least one sample is fully methylated and there is no methylation for at least another sample. To estimate the number of undetected DMRs, we calculate the expected proportion of SNP positions without two different homozygotes. This problem can be abstracted by a urns model containing spheres in three different colors. Two types of spheres with the probabilities p^2 and $q^2 = (1-p)^2$ represent the two different homozygous SNP states and the third type with the probability $2pq$ represents the heterozygous SNP state. Applying the inclusion-exclusion principle to the complementary event, the probability of obtaining such a position for n draws (samples) and p allele frequency is

$$P(p,q) = (1-p^2)^n + (1-q^2)^n - (2pq)^n.$$

With $q = 1 - p$ the equation is reduced to

$$\begin{aligned} P(p) &= (1-p^2)^n + (1-(1-p)^2)^n - (2p(1-p))^n \\ &= (1-p^2)^n + (-2p+p^2)^n - (2p-2p^2)^n. \end{aligned}$$

Assuming that a DMR is directly connected to exactly one SNP, we conclude that we can detect a DMR if we can detect the corresponding SNP. In general, for a given set of SNPs S and allele frequency p_s for SNPs $s \in S$ the expected SNP detection rate is given by

$$d = \frac{\sum_{s \in S} P(p_s)}{|S|}. \tag{7.1}$$

Since a DMR is connected to a single (random) SNP, we come to the conclusion that the detection rate of this DMR is d. Let D be the amount of DMRs identified. Even if we cannot determine the set of all available DMRs $R \supseteq D$, we can use the detection rate to estimate its size

$$|R| = \frac{|D|}{d}. \tag{7.2}$$

From over 300 Million publicly available SNPs from *dbSNP* (Sherry et al.,

2001) v143 we keep those with MAF < 0.05 and calculate a common detection rate of $d = 0.23$. Thus 77% of all existing common DMRs are undetected in our study.

7.2 DMR Characteristics

After validation, we classify the detected DMRs using different statistical values and metrics described below, which also serve as features for the currently developed methylome system (see Section 6.8). Biological explanations, interpretations, and hypotheses have been done by Elsa Leitao and Bernhard Horsthemke and are left out on purpose.

7.2.1 CGI Overlap

Unexpectedly, 29/157 DMRs overlap a CpG island (CGI). These regions with a high frequency of CpG sites stretches of DNA 500–1500 bp long and have a *observed-to-expected CpG ratio*

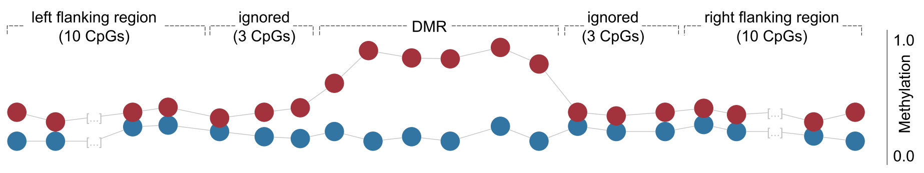
$$\frac{\text{number of CpG} \cdot \text{length of sequence}}{\text{number of C} \cdot \text{number of G}}$$

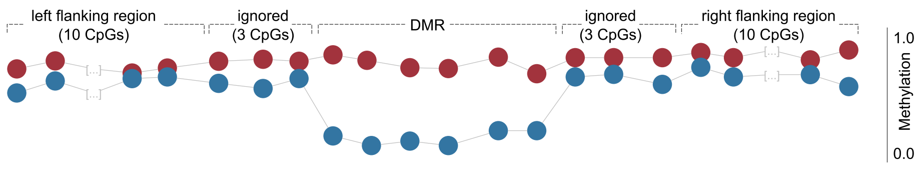
greater 0.6 (Gardiner-Garden and Frommer, 1987). CGIs are typically found at promoters and contain the 5′ end of the transcript (Cross and Bird, 1995) and are usually unmethylated and not affected by differential methylation.

In 24 of these CGI-DMRs all CpGs are within a CGI, in 4 cases there is a partial overlap with at least 50% of the CpGs belonging to a CGI, and in one case the CGI is within the DMR. In some cases, closely related DMRs affect the same CGI, probably because the DMR recognition algorithm splits a large DMR into two or more DMRs. A total of 19 CGIs overlap a DMR. Most of these CGIs are orphaned CGIs, i.e., they are not associated with a transcription start site (Illingworth et al., 2015).

7.2.2 Genomic Environment

We try to answer the question: Are interindividual DMRs typically surrounded by a highly methylated background and the DMR itself is unmethylated or vice versa. These different states are visualized in Figure 7.4. We compare the mean degree of methylation of each DMR with that of the flanking regions using all five encoders. To avoid uncertain DMR boundaries, we ignore three CpGs on both sides of each DMR and analyze the following 10 CpGs upstream and downstream. We call these 10 CpGs *right flanking region* and *left flocculating regions*. We observe that the mean degree of methylation of our DMRs is 0.49, which is close to the expected level of methylation when methylated and unmethylated alleles occur on

(a)


(b)


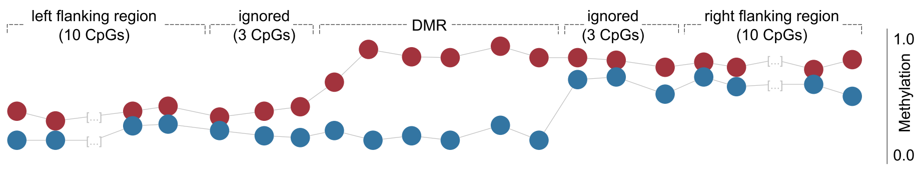
(c)


FIGURE 7.4: Examples of DMRs identified with the two synthetic methylomes (red and blue) with (a) low background methylation (7/157 DMRs) and (b) high background methylation (107/157 DMRs) (c) different background methylation on both sides of DMR (40/157 DMRs). Background methylation is measured on 10 CpGs flanking the upstream and downstream regions while ignoring 3 CpGs directly adjacent to the DMRs.

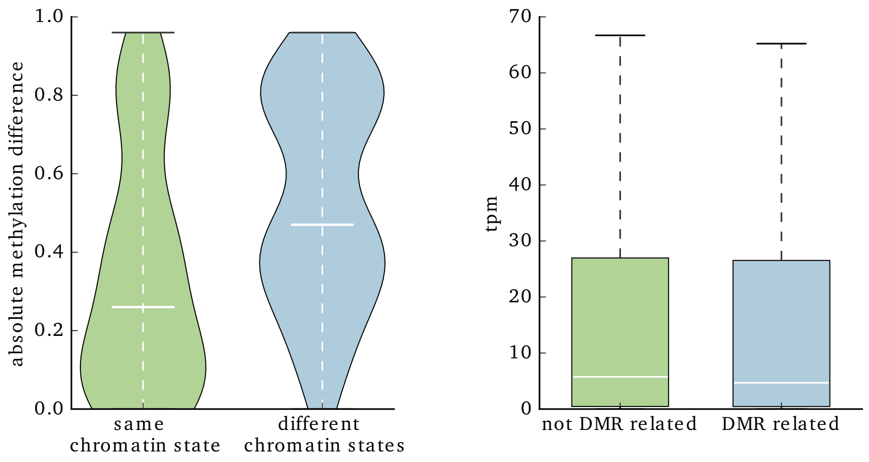
average in the five donors with a similar frequency. In contrast, both upstream and downstream flanking regions show a significantly higher degree of methylation at 0.72. The average of methylated and unmethylated alleles in the five donors is similar. In fact, 107/157 DMRs (68.2%) have an average methylation level lower than their two flanking regions (Figure 7.4) (b)), while only 7/157 (4.5%) have higher methylation (Figure 7.4) (a)). In 40 (25.5%) DMRs, methylation is higher in one flank region and lower in the other (Figure 7.4 (c)). For the remaining 3 DMRs, data is missing for one of their flanks. Thus, in most cases a *methylation valley* results from a causal genotype.

7.2.3 Chromatin States of the DMRs

Based on the combination of different histone marks in Hm03 and Hm05 monocytes, we applied with the help of ChromHMM (Ernst and Kellis, 2012), which segments a genome into 18 chromatin states (CS), and investigated whether certain chromatin states are over- or underrepresented in our DMRs. To test this, we estimate empirical p-values by simulating as follows:

1. Simulate 1 million datasets L, where $l \in L$ consists of 157 regions from non-repetitive segments of the DNA with similar distribution size compared to our DMRs D and covers at least 4 CpGs.

2. Identify overlaps of CS of Hm03 with each DMR.



(a) Chromatin states. (b) Gene expression.

FIGURE 7.5: (a) Absolute DMR differences of Hm03 and Hm05 separated for the groups of DMRs with same and different chromatin states. (b) Expressions of DMR related genes and not DMR related genes.

3. Identify overlaps of CS of Hm03 with each simulated region.
4. Compare the count of CS of the DMRs and each l

The empirical p-value for overrepresentation for chromatin state x is the fraction of L that have a higher count for x than our DMRs and underrepresentation the fraction of sets with a lower count. The analysis is performed analogously on Hm05. We find that in both data sets the states *1_TssA*, *5_Tx* and *17_ReprPCWk* were underrepresented and that *16_ReprPC* and *2_TssFlnk* or *4_TssFlnkD* were overrepresented.

Furthermore, we divide the absolute methylation differences between Hm03 and Hm05 into 1) a set for DMRs with different chromatin states in both donors and 2) a set for DMRs with the same chromatin states. We consider a state different for two donors when the cross between DMR and overlapping chromatin states is empty. As shown in Figure 7.5 (a) There are many DMRs with the same chromatin state in both donors that have no difference in DNA methylation (absolute methylation difference < 0.1), but there are very few DMRs with the same methylation with different chromatin states. Relative frequencies of DMRs with different chromatin states and methylation differences of 0.4 and 0.8 can be explained by homozygosity for one state in one donor and heterozygosity for the other, or homozygosity for opposite chromatin states. These results show that there is a correlation between DNA methylation and chromatin state. The significance test using the Wilcoxon-Rank sum test (Wilcoxon, 1945) gives a p-value of about 0.000023.

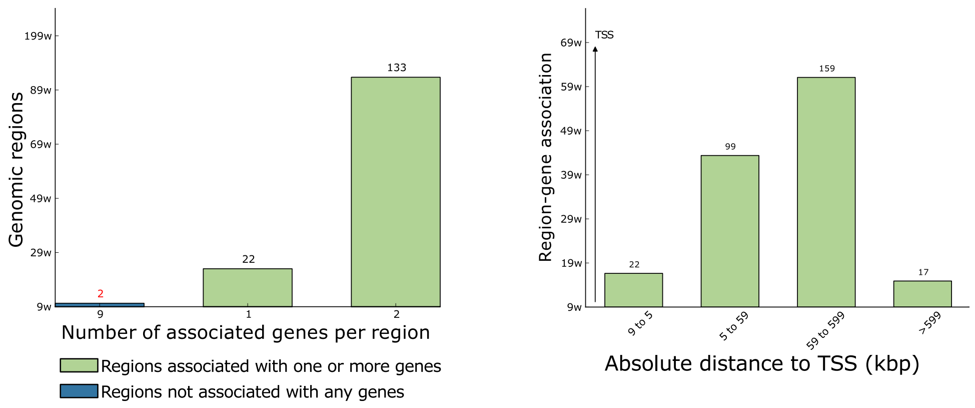


FIGURE 7.6: Two histograms of associated genes per DMR (left) and distance to their TSS (right) generated by GREAT (color modified).

7.2.4 Location of DMRs and Putative Target Genes

We use the Genomic Regions Enrichment of Annotations Tool (GREAT) (McLean et al., 2010) to predict DMR functions by analyzing the annotations of nearby genes under species Assembly GRCh37 with whole genome background and basal plus extension with standard parameters of 5 kilobases upstream, 1 kilobase downstream and up to 1000 kilobases distal. The identification of cis-regulating elements and their target genes by GREAT shows that 155/157 DMRs are associated with at least one gene and that in most cases they are far away (Figure 7.6). A total of 240 different genes were identified. The expression levels of these genes do not differ from those not associated with DMRs, with a p-value of 0.45 in the Wilcoxon-Rank sum test. As shown in Figure 7.5 (b), there is no correlation between differences in gene expression and differences in methylation in Hm03 and Hm05. There was also no significant accumulation of GO conditions.

7.2.5 Methylation Level Distribution

A more detailed investigation shows that the CpGs of a particular sample in these regions are either nearly unmethylated, fully methylated or about 50% methylated and 50% unmethylated. For semi-methylated samples that we observe, most reads are either fully methylated or non-methylated and rarely contain CpGs of both states. Figure 7.7 shows an exemplary region with methylation data of the five donors at read level. We conclude that methylation in these regions is likely to be allele-specific, with one of two alleles methylated and the other non-methylated, and investigate this below.

7.3 DMR-to-SNP correlation

Allele-specific DNA methylation indicates an inheritance mechanism. Since methylation states are not directly inherited, we assume that this

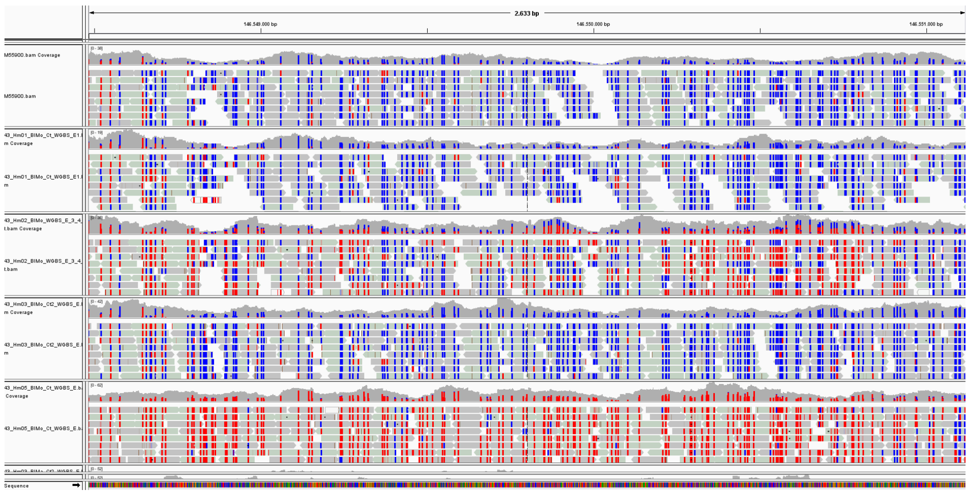


FIGURE 7.7: IGV screenshot of a detected interindividual DMR. Each of the five separated rows shows reads (bars) for one of the donors with methylated (red) and unmethylated (blue) CpGs.

type of methylation is probably caused by inherited single nucleotide polymorphisms (SNPs) near the differentially methylated region and therefore depends on the genotype. To prove this hypothesis, we genotyped the five donors with the 2.5 million SNPs Omni2.5Exome Bead Array produced by Illumina.

To identify a correlation between SNPs and methylation states, we have developed a correlation value. As mentioned above, we expected and observed allele-specific methylation levels close to 0.0, close to 1.0, or about 0.5. We assume three possible classes: *fully-methylated* (both alleles methylated), *halmethylated* (one of the two alleles methylated) and *unmethylated* (both alleles unmethylated) for this epigenetic genotype. To compare these epigenotypes with SNP genotypes, we need to classify the degree of methylation of each sample for each DMR. Due to inaccurate DMR boundaries, limited sequence coverage, and noise, measurements may deviate from this expectation. We avoid fixed thresholds for class allocation by calculating the posterior probabilities of mean DMR methylation levels falling into each of the classes as follows. We consider the empirical distribution (histogram) of $157 \cdot 5 = 768$ (#DMRs · #samples) core methylation level $\mu_i(d)$ (see Chapter 3.3) of each sample i and DMR d containing data from all three classes. This empirical distribution can be broken down into a three-component mixture of beta distributions (see Section 4). We used the `betamix` software to robustly fit a three-component beta-mix model to the observed histogram of nuclear methylation values. This is shown in Figure 7.8.

Let α_k and β_k be the beta distribution parameters for beta distribution $B_{\alpha,\beta}(x)$ and π_k the mixture coefficient of component $k \in \{AA, AB, BB\}$ after fitting. For a single sample, DMR with core methylation level μ and SNP genotype $g \in \{AA, AB, BB\}$, the posterior probability of g given μ is given by

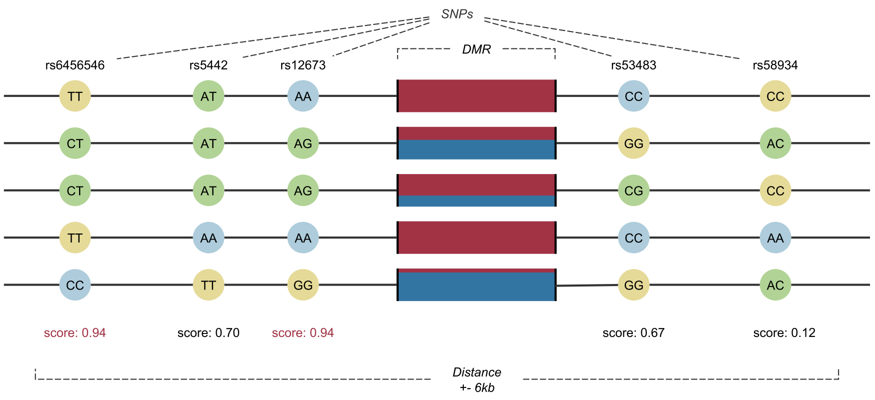


FIGURE 7.8: Exemplary visualization of the assignment of a correlation value for each DMR within a certain range (here 6kb). Each line represents a string from one of five samples. For each sample, methylation for DMR is represented as the ratio of methylated (red) to unmethylated (blue) CpGs. The genotypes of the SNPs considered are homozygous (blue), heterozygous (green) or homozygous for the alternative base (yellow). The two best correlation values (marked red) form an SNP group.

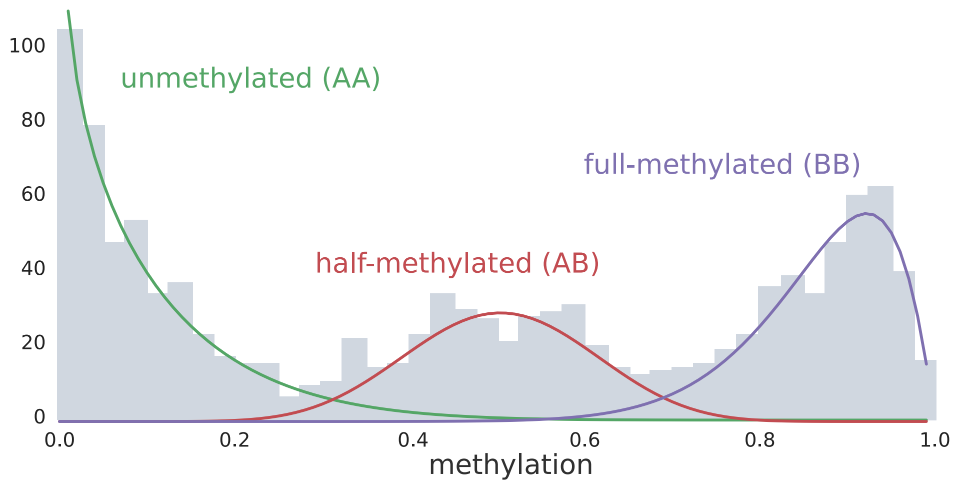


FIGURE 7.9: Histogram of the 768 nuclear methylation values, adapted to three beta distributions, showing the three different states *full-* (green), *halb-* (red) and *unmethylated* (purple).

$$L(g,\mu) = \frac{\pi_g \, B_{\alpha_g,\beta_g}(\mu)}{\sum_k \pi_k \, B_{\alpha_k,\beta_k}(\mu)}.$$

Furthermore let $g_i(s) \in \{AA, AB, BB\}$ be the genotype and $\mu_i(d)$ the core methylation for a DMR d, sample i and SNP s. The posterior probability

$$\text{score}(s,d) = \prod_{\text{samples } i} L(g_i(s), m_i(d)).$$

is given by the product of the single posterior probabilities over all samples. We use this posterior probability as a score to assess whether d and s are co-varying. Scores are calculated for each DMR and SNP within a range of ±6 kb of the DMR's location. For $n = 5$ samples, we used a SNP-to-SNP correlation of 0.9 obtained from the `HaploReg` database (Ward and Kellis, 2012) as threshold to call a SNP correlated to a DMR. 82/157 (52%) of the DMRs have a methylation level that is correlated with the genotype of at least one nearby SNP. In 21/157 DMRs that SNP is located within the corresponding DMR and in 18/157 its location is in the distance < 200 bp from the corresponding DMR border.

7.4 Representative SNPs

Results such as those described above must be verified experimentally. We have developed an approach for the SNPs that need to be verified for our DMR/SNP correlation.

Genotypes of two SNPs often correlate up to a perfect correlation of 1.0. This phenomenon is described by the so-called binding imbalance (LD), a non-random association of alleles at different sites in a given population (Slatkin, 2008). Loci are said to be in linkage disequilibrium if the frequency of association of their different alleles is higher or lower than expected if the loci were independently and randomly associated. LD is influenced by many factors, including selection, recombination rate, mutation rate, genetic drift, mating system, population structure, and genetic linkage. The verification of an SNP indirectly verifies all correlated SNPs of the same LD. In order to minimize the number of SNPs to be verified (and the verification costs), we first formulate the situation as a dominant set problem.

Dominating Set Problem In graph theory, a dominanting set for a graph $G = (V, E)$ is a subset $D \subseteq V$, so that any node not contained in D is adjacent to at least one element of D. The *minimum dominating set* (MDS) problem is to find the smallest dominant set. Figure 7.10 shows three different dominating sets for the same graph. The problem is NP-complete (McDonald, 2014), but there is an approximation algorithm, e.g. as part of the Python graph framework NetworkX (Hagberg et al., 2008).

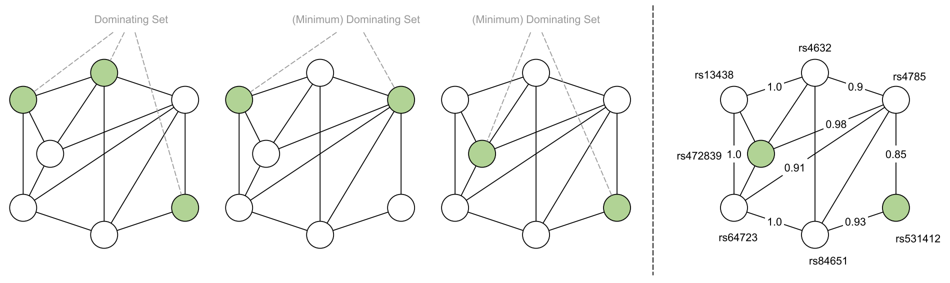


FIGURE 7.10: Illustration of the minimum dominant set problem. The same diagram is shown three times (left), each with a different dominating set (green). The dominant sets on the second and third position are *minimal* since there is no smaller dominant set than size two. The right graph shows the same example with annotation as it would be constructed from the DMR correlation and the *HapMap* data. Each node named by a rsid represents an SNP with DMR correlation. Each edge represents an existing SNP to SNP genotype correlation and is provided with this coefficient.

We'll create a graph based on *SNP to SNP* genotype correlations obtained from *HapMap* - publicly available precalculated LD information with correlation coefficients provided by the *The International HapMap Project*. (International HapMap Consortium, 2003).

For each set of SNPs S and DMR d selected for verification, we add a v_s node for each $s \in S$. An edge (v_{s_1}, v_{s_2}) exists between two nodes if and only if the genotype correlation between s_1 and s_2 is above a selected threshold, e.g. > 0.8. The goal is to select the minimum set of representatives so that every second $s \in S$ is correlated with at least one representative. The MDS solution of this graph provides a minimum number of SNPs that must be verified by targeted sequencing and that also indirectly verify S. To solve this problem we used the approximation algorithm of NetworkX, which is an implementation of an algorithm of Vazirani (2001). The approximation algorithm always led to a correct MDS solution for our small graphs. Another way to solve the MDS is an ILP with the indicator variables
in 0,1
for each node and a sum that ensures that the node itself or one of its neighbors is selected. The target function to be minimized is then the sum of the indicator variables. We finally decided to check *all* SNPs instead, but the idea might still be useful in future studies.

7.5 Whole Genome Association

A correlation analysis based on five monocytes leads to a low level of evidence. The *Heinz Nixdorf Recall Study* (Erbel et al., 2012) provided us with 450k and SNP array data from 1131 donors to increase the significance of the study. The tissue of these samples was blood (whole blood) consisting of mixed cell types. The table 7.1 shows the distribution of different cell types in whole blood tissue. We used isolated monocytes for our analysis,

Name	cells per μl blood	percentage
Erythrocytes	4.5 - 5.5 Mio.	$\sim$ 94%
Leucocytes	4.000 - 11.000	$\sim$ 0.1%
| - Granulocytes		
| | - Neutrophiles	2.500 - 7.500	$\sim$ 0.09%
| | - Eosinophiles	40 - 400	$\sim$ 0.004%
| | - Vasophiles	10 - 100	$\sim$ 0.001 %
| - Lymphocytes	1.500 - 3.500	$\sim$ 0.05 %
| - Monocytes	200 - 800	$\sim$ 0.01 %
Thrombocytes	300000	$\sim$ 5%

TABLE 7.1: The distribution of different cell types in whole blood tissue.

which make up only a small part (about 0.01%) of the whole blood and our detected DMRs may be monocyte exclusive and may not occur in this tissue.

We investigate the possibility of using whole blood material provided. A correlation coefficient of $r > 0.9$ for the correlation of methylation data of pure monocytes of six donors with their whole blood samples proofs that we can. Figure 7.11 shows this correlation for each of the six donors.

450k methylation arrays cover the status of 485,764 CpG of the 30 million CpGs of the human genome. Our goal is to indirectly correlate our DMRs with SNPs by selecting single 450k CpGs that represent a complete DMR. After using RnBeads (Assenov et al., 2014) for normalization by SWAN (Maksimovic et al., 2012) and exporting the CpG beta values for all 1131 donors, we select representative CpGs by searching for DMRs that overlap 450k CpG probes. Only 51 CpGs overlap with one DMR and 30/157 (19%) DMRs contain at least one 450k CpG. One implication of this is that 450k arrays miss a large part of the DNA methylation variation and are not capable of recognizing (short) interindividual DMRs. Even when a differential analysis is performed on individual CpGs, only a fraction of about 20% of the DMRs present is obtained at maximum. The degree of methylation of the covered CpGs serves as a representative for whole DMRs. Figure 7.12 shows histograms of six examples of such CpGs. We observe the similar multi-beta distribution of methylation for most of the representative CpGs, which is an expected result. Instead of the mean DMR methylation, we want to correlate the methylation values for each representative with the genotypes of SNPs.

The SNP array data were generated with three different SNP array types: *Omni1_Quad_v1* (334 subjects), *OmniExpress_12v1.0* (627 subjects) and *OmniExpress_12v1.1* (170 subjects). We filter each array separately by removing SNPs that did not pass the *Hardy-Weinberg* test at a significance limit of 0.001, a small allele frequency of less than 0.01, or a missing rate of more than 0.1 reported by `plink` v1.07 (Purcell et al., 2007). Because each array type covers different SNPs, we merge the arrays by `plink` and filter (again with `plink`) using the parameters described above. The final set contains 600,000 SNPs with data from each array serving as genotypes for the GWASs.

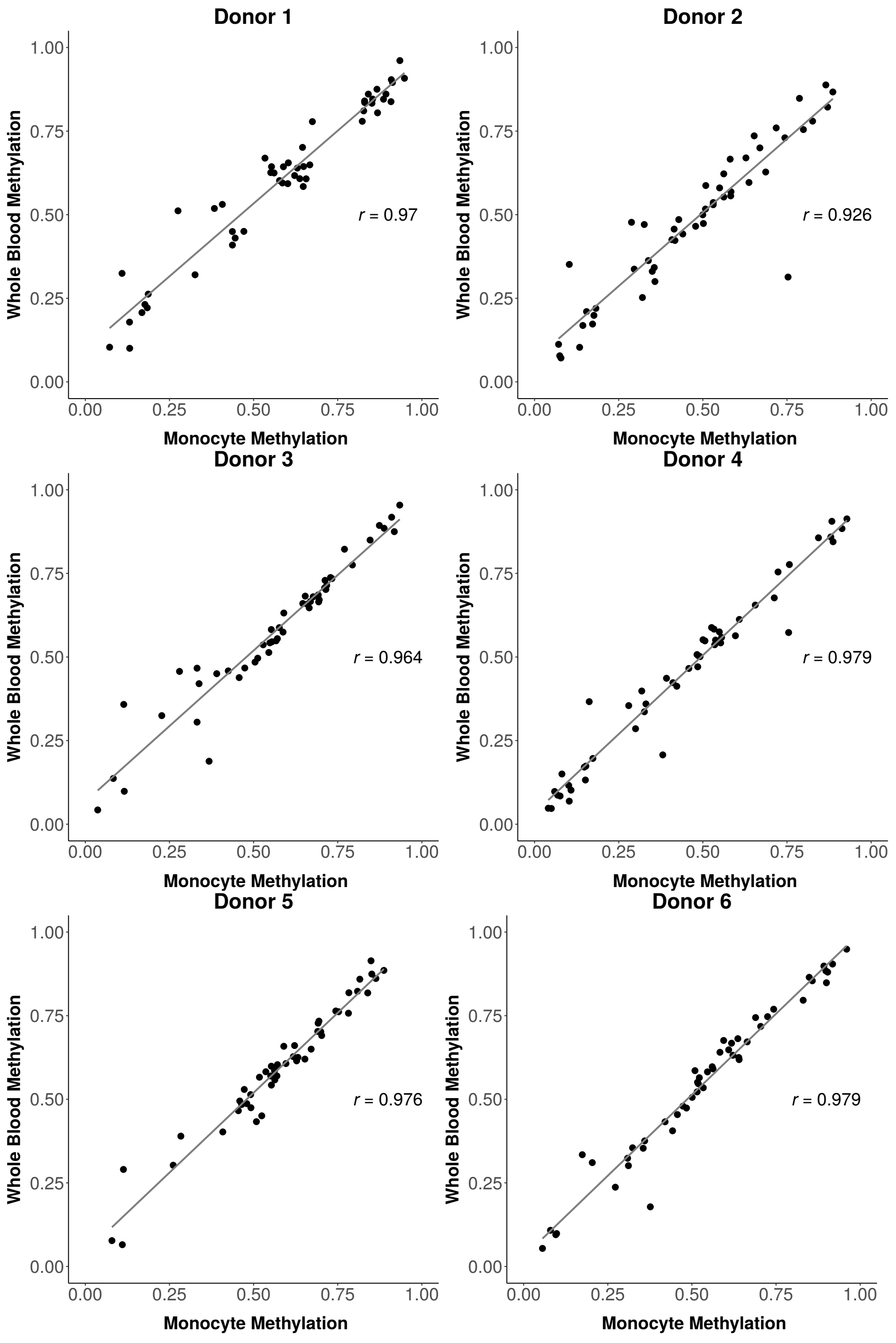


FIGURE 7.11: Scatterplot of methylation in monocytes and whole blood. Each dot represents one of our detected DMRs. The x-value of each dot represents the mean methylation of the DMR in the monocytes cell type; the y-value the mean methylation in whole-blood for the same donor. Additionally the correlation coefficient (pearsons r) for each donor is annotated.

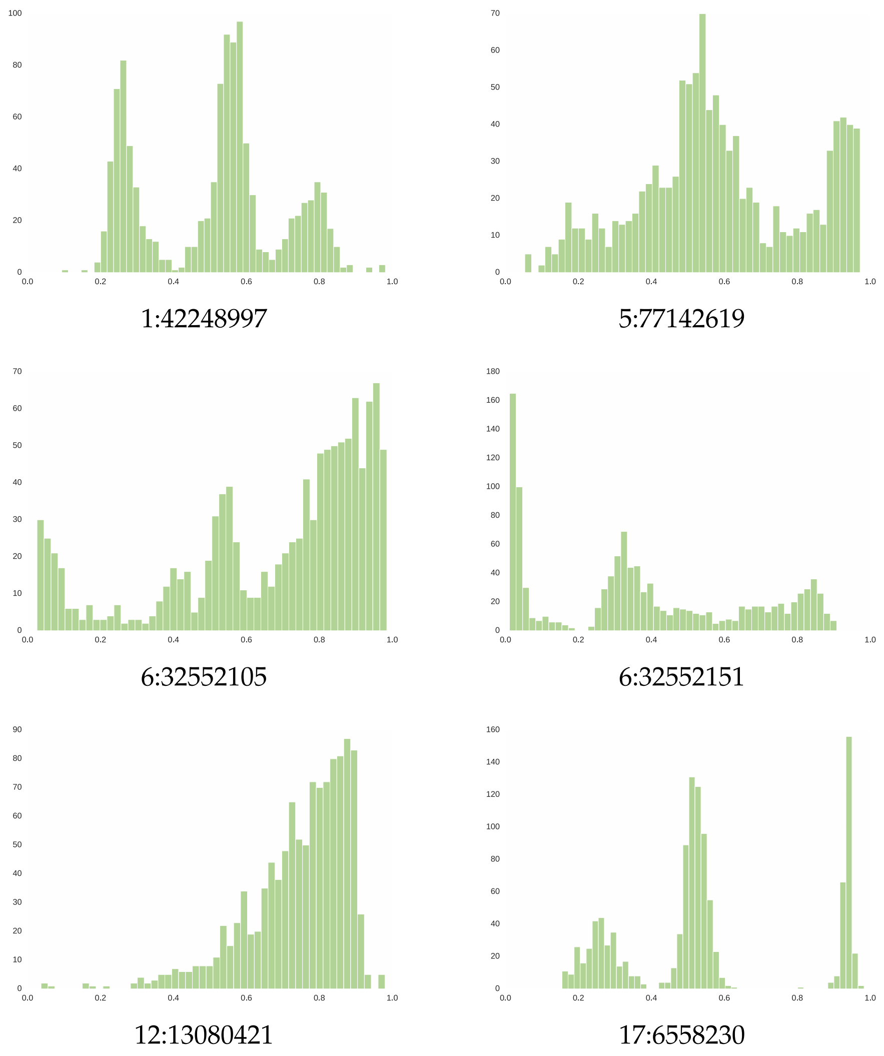
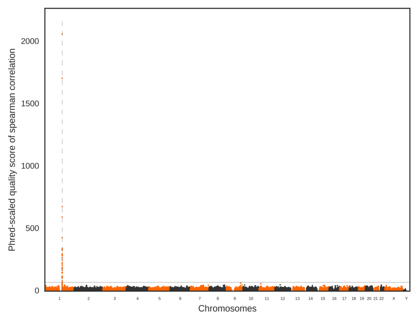


FIGURE 7.12: Six example histograms of different shapes generated from 1131 (450k) methylation values in six selected single CpG. The subcaption of the six plots refer to the chromosomal position of the CpG.

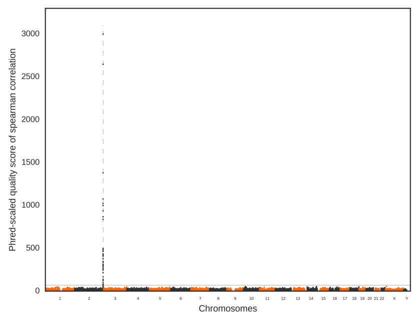
We perform 51 whole genome association studies (GWAS) for each of the 51 selected CpGs for each available SNP, based on the methylation and genotype values of the 1131 donors. Methylation and genotype values are not necessarily normally distributed. We, therefore, apply the Spearman test (McDonald, 2014) and set the p-value threshold to the standard for Common-Variant GWAS of 5×10^{-8}(Fadista et al., 2016).

As expected, in 47/51 cases we can observe correlating SNPs near the CpG position. Figure 7.13 shows Manhattan plots of six selected examples of this Spearman correlation. The SNP with the highest correlation value for each CpG c is called *lead SNP* L_c. It has the highest probability of causing the DMR methylation differences. For the CpG in DMR94 on chromosome 12, there is a correlation peak at the CpG position ($p = 1.59 \times 10^{-15}$), but the lead SNP is on chromosome 19 ($p = 1.40 \times 10^{-41}$).

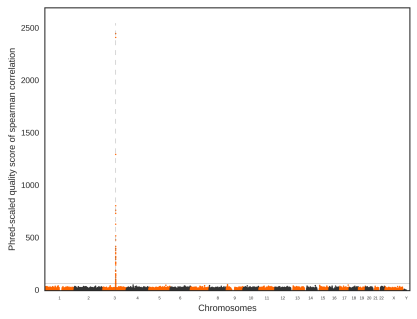
DMR53 and DMR94 (not shown) are two special cases, where, the DMR



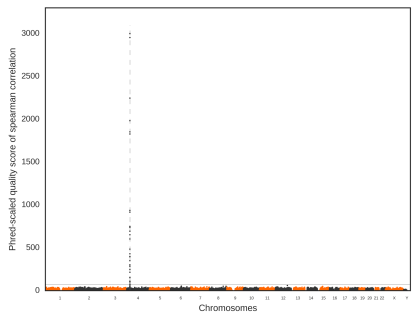
(a) 1:146550796



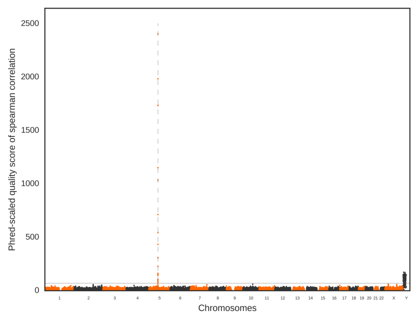
(b) 2:2400861789



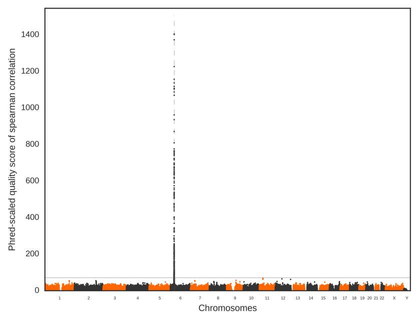
(c) 3:108125523



(d) 4:26454190



(e) 5:77142695



(f) 6:32552016

FIGURE 7.13: Manhattan plots of six exemplary *CpG-to-SNP* correlations of a total of 51 association tests. Each plot shows a complete genome association study for the representative CpG methylation value of a single DMR on 600,000 SNPs. Each point in each plot represents a single *CpG-to-SNP* correlation with the x coordinate as the genomic position and their coordinate of the Phred-scaled p-value of the Spearman correlation. The test is performed on SNP genotypes and methylation values of 1131 sample pairs from the *Heinz-Nixdorf study*. The vertical dashed line in each plot shows the DMR position. Chromosomes are alternately colored black and orange. The correlation peak, the *lead SNP*, is directly adjacent to the CpG position and shows that there is almost one SNP responsible for the methylation state of an entire region (the DMR).

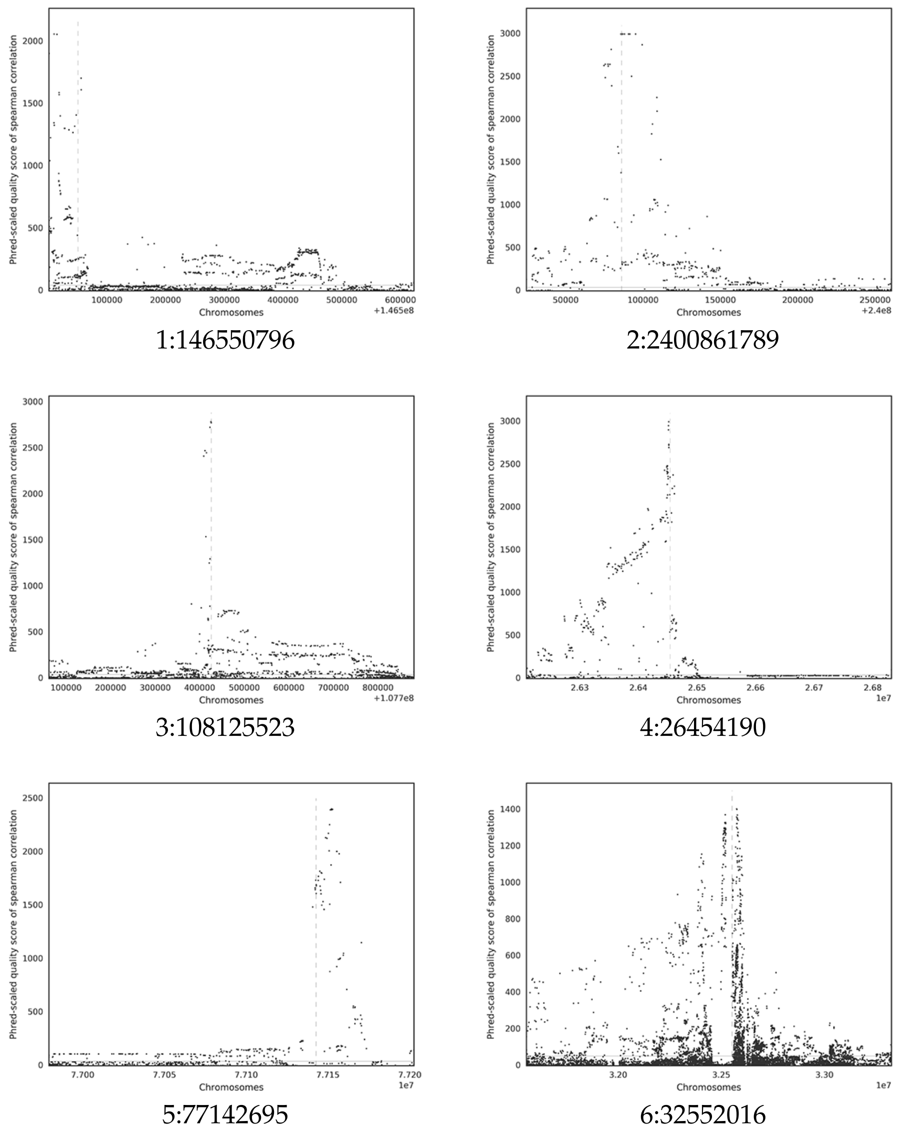


FIGURE 7.14: Six examples of the imputed regions. Each dot represents the same information as in figure 7.13

chromosome and the lead SNP chromosome differ. In these cases, we cannot find any evidence of misannotation or cross-hybridization of the array probes. Besides, we find that these lead SNPs overlap with genes coding for KRAB zinc finger transcription factors (ZNF573 for lead SNP of DMR94, ZNF92 for lead SNP of DMR53). Extremely low p-values at ZNF573 and ZNF92 indicate a trans-active effect [1].

Since our uniform SNP dataset M consists of only 600,000 SNPs due to the merge process, we compensate for missing information by assigning the dataset of SNPs as follows. For each CpG, a window is selected so that all significantly correlated SNPs and the CpGs are in that window, with a maximum window size of 1Mbp. We use `impute2` (Howie et al., 2009) to calculate genotype information from each SNP $s \notin M$ in this window. This

[1]gene superposition and interpretation by Bernhard Horsthemke and Elsa Leitao

process makes use of *SNP-to-SNP* correlation information from *HapMap* to estimate the most likely genotype of s. Manhattan plots of the imputed regions (Figure 7.14) do not share a common pattern of significantly correlated SNPs to the CpG.

TFBS In and Around the DMRs

We analyze whether lead SNPs and highly correlated SNPs (total $n = 501$) can influence the binding sites for transcription factors (TFBS). The analysis of SNP annotation data from the database `HaploReg` showed that 23% of known protein binding events (Encode ChIPseq data, Landt et al., 2012) occur within our DMRs or <100 bp away (Fig. S14). The remaining events occur over a region ±57 kb away from the DMRs. The five best proteins that bind to DMR or in close proximity (<100 bp) are CTCF, CMYC, CEBPB, RAD21, and SMC3.

The TRANSFAC (Matys et al., 2006) Analysis showed that SNP regions are enriched for CREB group, NF-1, Sp100, and CTCF binding motifs, and further analysis of their `HaploReg` database notes revealed that most of them are likely to change regulatory motifs.

7.6 Summary

This chapter mainly shows methods for the detection and analysis of novel regions with common interindividual DNA methylation differences in human monocytes. Our study supports the observation that genetic variations in cis can cause allelic DNA methylation differences. While some parts, such as the detection of DMRs, require special treatment (i.e., the production of synthetic methylomes), most annotations and discoveries are universally applicable to all DMR discoveries. Therefore, this chapter also serves as a blueprint for future analyses where an automated pipeline performs these or similar annotations. A graphical system (currently under development) based on the `Eagle` structure can provide annotations to any given or generated list of DMRs.

Chapter 8

Conclusion

In this work, we presented methods to manage all phases of methylome generation and analysis. In Chapter 2 we offer a way to improve sequencing protocols by sequencing subsamples and estimating the duplication rate for samples with higher coverage, one of the most important quality parameters for NGS data. We explained the general approach of wildcard and three-letter aligners for dealing with bisulfite converted sequencing data and presented in Chapter 3 `camel`, a resource-conserving tool for performing methylation calls and downstream methylome analyses. In Chapter 4 we have developed an alternative for the EM algorithm, the *iterated method of moments*, which is capable of robustly processing data points of 0 and 1 for the special case of beta distributions. This allows us to adapt the parameters of beta mixture models to a distribution of measured methylation levels where these values are widespread. Chapter 5 provides two workflows, both based on the workflow management system `Snakemake`. `Ape` can perform all necessary steps to call variants from reads, that are located in specific target regions (e.g., the exome). The more complex `moquito` handles WGBS data with an increased amount of data that can even be sequenced across multiple tracks. Variants of `ape` can be used by variant analysis systems, such as `eagle`, which we introduced in Chapter 6. Finally, in Chapter 7 we provided a method to detect interindividual methylation between multiple donors, and used the methods of the previous chapters to perform a downstream analysis of the found DMRs and their variants.

8.1 Open Problems And Ideas

8.1.1 Duplication Detection

Neural networks are an alternative to the method presented for predicting the duplication rate (Chapter 2). We can encode a document vector and an upsampling rate as an input to get a single output value, which should be possible with Multilayer Perceptron (MLP). If we use many layers for the network, we are in the area of deep learning (DL), which has recently shown superior performance in several bioinformatics tasks such as medical imaging (Suzuki, 2017), prediction of gene expression from histone modifications (Singh et al., 2016) or sequence specificities of DNA and

RNA binding proteins (Lanchantin et al., 2016). Training data can be obtained by downsampling artificial or real data and different rates.

8.1.2 Remove Incompletly Converted Reads

A problem with WGBS are fragments that are not affected by bisulfite, which causes noise and a tendency for increased methylation values. These values can be identified by searching for unconverted cytosines outside the CpG context, where cytosine is expected to be unmethylated. This in turn can be implemented in the methylation level call, e.g. by `camel`. (Chapter 3.2). Another way is to add one additional step after the read-alignment, set the Quality-Control-fail (QC flag) flag in those that display such unexpectedly unconverted cytosines, and ignore the reads marked with this flag when calling methylation. This approach is less complicated because the reads of the same fragment are already paired. In both cases, any methylated cytosine outside a CpG context on one of the two reads indicates conversion problems. Also, sequencing errors can also lead to these cytosines. This must be taken into account, for example by using the base quality.

8.1.3 DMR Detection

Many different approaches for DMRs detection have been developed:

- `QDMR` (Zhang et al., 2011)
- `bumphunter` (Jaffe et al., 2012)
- `bsmooth` (Hansen et al., 2012)
- `CpG_MPs` (Su et al., 2013)
- `MethylPurify` (Zheng et al., 2014)
- `ComMet` (Saito et al., 2014)
- `swDMR` (Wang et al., 2015)
- `DSS-single` (Wu et al., 2015)
- `GetisDMR` (Wen et al., 2016)
- `MethylAction` (Bhasin et al., 2016)
- `metilene` (Jühling et al., 2015)
- ...

We recently applied `bsmooth` and `metilene` to two different sperm methylomes (unpublished) and found that the DMRs returned by both tools did not overlap. We also discover that the regions identified by `bsmooth` are smaller than those of `metilene`. Selected DMRs from both sets could be verified experimentally, and finally, we assume that both tools provide true positive and false negative results (low sensitivity). A comparison of all available DMR detection tools to identify their specific properties, e.g., DMR length, shapes or boundaries (see Chapter 3.3), detection rate and p-values. It would be advantageous for such a study to simulate DMRs of different length and shape between a variable number of methylomes. Previous work, e.g., (Wang et al., 2015), implement a simple DMR simulation based on a bimodal distribution and compare their obtained regions with those of two other tools (`ComMet` and `CpG_MPs`). Again, each of the three applied tools identifies regions with unique properties and many of them are exclusive. A comprehensive and fair study with all tools requires a realistic model. To simulate a methylome, a simulator must take into account the following: CpG methylation distribution that can follow a 3-component beta distribution (see Chapter 7), dependence on neighboring CpGs and noise from biological variance and finite sample size (coverage). The idea is to use the high-quality WGBS samples produced by DEEP to estimate all parameters of these distributions. In a second step, we apply all available DMR detection tools and create a common set of DMRs D and create a DMR simulation model by estimating parameters such as the length and shape of all DMRs captured. We then simulate biological replicas of any coverage, including DMRs (with different parameters), to validate and compare the tools and identify possible detection properties and problems.

Another idea is to train a convolutional neural network (CNN) to detect differentially methylated CpGs. The network is trained either on CpGs that are covered by D and receive a generalized method that covers all other approaches or on simulated DMRs to obtain an algorithm that is independent of other tools. CNNs are specifically used to detect patterns in matrices, such as images, and a similar approach has recently been used to identify SNPs (Poplin et al., 2018). The basic idea is to decode all relevant information, such as bases, distances, and neighborhood of each CpG, through normalized matrices. A trained network can then recognize a DMC not only from the CpG pileup but also from its context through a sliding window. Continuous DMCs then build DMRs.

8.1.4 Methylation-to-SNP Correlation

Other studies may also be influenced by the p-values of the *methylation to SNP* genotype Spearman correlations described in Chapter 7.3. We are currently designing a database containing p-values for all 600,000 SNPs and approximately 500,000 methylation levels of the 450k array and 1131 donors. The use of 32bit floats would result in $600,000 \cdot 500,000 \cdot 32$ bit $=$ 1.2 TB p-values. By calculating only correlations between methylation and genotype probes of the same chromosome, we would drastically reduce the number of p-values. We also miss interesting correlations where SNPs influence the degree of methylation on different chromosomes. Another possibility is to calculate all values but store only those below the general GWAS threshold. In both cases, it is possible to store the homogeneous data in *hdf5*. A graphical interface can then be used to enter DMRs and return associated (ranges of) SNPs or vice versa. It may be necessary to include an SNP imputation algorithm that covers all available SNPs. Furthermore, we want to investigate whether correlated SNPs located on different chromosomes are always located within regions associated with methylation regulation.

8.2 Acknowledgments

I acknowledge funding from the Federal Ministry of Education and Research (BMBF) under the Project Number 01KU1216 (Deutschemes Epigenome Programm, DEEP).

Chapter 9

Appendix

9.1 Contribution to co-authored articles

Results in this work were obtained in collaboration with other researchers. As advisor Sven Rahmann supports in all phases of research.

Chapter 2 is based on my conference talk on the *German Conference on Bioinformatics 2015* and is published as proceedings:

Christopher Schröder and Sven Rahmann (2015). "Efficient duplicate rate estimation from subsamples of sequencing libraries". *PeerJ PrePrints*. DOI: `10.7287/peerj.preprints.1298v2`

I developed the family size reduction by hypergeometric distributions and the core LP to estimate the supersample duplication rate and evaluation and comparison to other tools. Sven Rahmann developed the extension for estimating the 95% quantiles and the maximum k. All writings and figures has been done by Sven Rahmnann and me.

The method parameter estimation of beta mixtures (Chapter 4) bases on

Christopher Schröder and Sven Rahmann (Aug. 2017). "A hybrid parameter estimation algorithm for beta mixtures and applications to methylation state classification". *Algorithms for Molecular Biology* 12, 21–33. DOI: `10.1186/s13015-017-0112-1`

and has been equally developed by Sven Rahmann and me. He performed the parameter estimation evaluation and did most of the writings. I performed the component estimation evaluation and the implementation of the core algorithm and stop criterion and submitted it to Bioconda for

Björn Grüning, Ryan Dale, Andreas Sjödin, Jillian Rowe, Brad A. Chapman, Christopher H. Tomkins-Tinch, Renan Valieris, The Bioconda Team, and Johannes Köster (Oct. 2017). "Bioconda: A sustainable and comprehensive software distribution for the life sciences". *bioRxiv*, 207092. DOI: `10.1101/207092`

as part of the Bioconda Team.

I implemented the system core and most of the features of `Eagle` described in Chapter 6. Some graphical representations in `Eagle` were implemented by Christoph Stahl and Sebastian Venier and structural variants as well as loss of heterozygosity by Felix Mölder. The purity calculation is based on an implementation by Johannes Köster, who also contributed written material, as well as Felix Mölder and Till Hartmann. A publication of `Eagle` is currently in revision with and the evaluation has been performed in comparison to results of

> Alexander Schramm, Johannes Köster, Yassen Assenov, Kristina Althoff, Martin Peifer, Ellen Mahlow, Andrea Odersky, Daniela Beisser, Corinna Ernst, Anton G. Henssen, Harald Stephan, Christopher Schröder, Lukas Heukamp, Anne Engesser, Yvonne Kahlert, Jessica Theissen, Barbara Hero, Frederik Roels, Janine Altmüller, Peter Nürnberg, Kathy Astrahantseff, Christian Gloeckner, Katleen De Preter, Christoph Plass, Sangkyun Lee, Holger N. Lode, Kai-Oliver Henrich, Moritz Gartlgruber, Frank Speleman, Peter Schmezer, Frank Westermann, Sven Rahmann, Matthias Fischer, Angelika Eggert, and Johannes H. Schulte (Aug. 2015). "Mutational dynamics between primary and relapse neuroblastomas". eng. *Nature Genetics* 47.8, 872–877. DOI: `10.1038/ng.3349`

where I analyzed the 450k methylation arrays.

I implemented both pipelines (`ape` and `mosquito`) described in Chapter 5 and I processed the exome data of

> Matthias Begemann, Faisal I Rezwan, Jasmin Beygo, Louise E Docherty, Julia Kolarova, Christopher Schroeder, Karin Buiting, Kamal Chokkalingam, Franziska Degenhardt, Emma L Wakeling, Stephanie Kleinle, Daniela González Fassrainer, Barbara Oehl-Jaschkowitz, Claire L S Turner, Michal Patalan, Maria Gizewska, Gerhard Binder, Can Thi Bich Ngoc, Vu Chi Dung, Sarju G Mehta, Gareth Baynam, Julian P Hamilton-Shield, Sara Aljareh, Oluwakemi Lokulo-Sodipe, Rachel Horton, Reiner Siebert, Miriam Elbracht, Isabel Karen Temple, Thomas Eggermann, and Deborah J G Mackay (July 2018). "Maternal variants in NLRP and other maternal effect proteins are associated with multilocus imprinting disturbance in offspring". *Journal of Medical Genetics* 55.7, 497–504. DOI: `10.1136/jmedgenet-2017-105190`

by `Ape` and provided them in `Eagle`.

Chapter 7 is based on the publication

> Christopher Schröder, Elsa Leitão, Stefan Wallner, Gerd Schmitz, Ludger Klein-Hitpass, Anupam Sinha, Karl-Heinz Jöckel, Stefanie Heilmann-Heimbach, Per Hoffmann, Markus M. Nöthen, Michael Steffens, Peter Ebert, Sven Rahmann, and Bernhard Horsthemke (July 2017). "Regions of common inter-individual DNA methylation differences in human monocytes: genetic basis and potential function". *Epigenetics & Chromatin* 10, 37–55. DOI: `10.1186/s13072-017-0144-2`

which itself emerged from an observation in

Stefan Wallner, Christopher Schröder, Elsa Leitão, Tea Berulava, Claudia Haak, Daniela Beißer, Sven Rahmann, Andreas S. Richter, Thomas Manke, Ulrike Bönisch, Laura Arrigoni, Sebastian Fröhler, Filippos Klironomos, Wei Chen, Nikolaus Rajewsky, Fabian Müller, Peter Ebert, Thomas Lengauer, Matthias Barann, Philip Rosenstiel, Gilles Gasparoni, Karl Nordström, Jörn Walter, Benedikt Brors, Gideon Zipprich, Bärbel Felder, Ludger Klein-Hitpass, Corinna Attenberger, Gerd Schmitz, and Bernhard Horsthemke (2016). "Epigenetic dynamics of monocyte-to-macrophage differentiation". ENG. *Epigenetics & Chromatin* 9, 33–50. DOI: `10.1186/s13072-016-0079-z`

I processed that methylation data via `mosquito` and performed the bioinformatic analysis for both publication.

The following publications did not contribute directly to this work, but they did give me an insight into biological processes and helped me to expand my bioinformatics skills.

Katrin Rademacher, Christopher Schröder, Deniz Kanber, Ludger Klein-Hitpass, Stefan Wallner, Michael Zeschnigk, and Bernhard Horsthemke (July 2014). "Evolutionary Origin and Methylation Status of Human Intronic CpG Islands that Are Not Present in Mouse". *Genome Biology and Evolution* 6.7, 1579–1588. DOI: `10.1093/gbe/evu125`

Christopher Schröder, Nina Hesse, Johannes Köster, and Sven Rahmann (2016b). *Methods for discovering differentially methylated regions from whole genome bisulfite sequencing data (and keeping track of the process)*. GMDS Workshop: From Biomedical Informatics to Medical Bioinformatics and back HEC, München

Christopher Schröder, Johannes Köster, and Sven Rahmann (2016a). *Interactivity vs. Reproducibility in High-Throughput Data Analysis*. German Bioinformatics Core Units Workshop, BIH, GCB Satellite, Berlin

Christopher Schröder and Sven Rahmann (2014). *Efficient duplicate rate estimation from subsamples of sequencing libraries*. German Conference on Bioinformatics, Dortmund

Nina Hesse, Christopher Schröder, and Sven Rahmann (2014). *An optimization approach to detect differentially methylated regions from Whole Genome Bisulfite Sequencing data*. German Conference on Bioinformatics, Dortmund

Leitão Elsa, Stefan Wallner, Christopher Schröder, Gerd Schmitz, and Bernhard Horsthemke (2014). *Epigenetic dynamics of monocyte to macrophage differentiation*. Gesellschaft für Humangenetik, Lübeck

9.2 Data

Data used in each chapters is available at the corresponding puplications. The DMRs for the comparions in Section 3.6 has been made public under: https://bitbucket.org/christopherschroeder/disseration_daten

Bibliography

Alberts, Bruce., John. Wilson, and Tim. Hunt (2008). *Molecular biology of the cell*. Garland Science.

Ameur, A., I. Bunikis, S. Enroth, and U. Gyllensten (Oct. 2014). "CanvasDB: a local database infrastructure for analysis of targeted- and whole genome re-sequencing projects". *Database* 2014.bau098. DOI: 10.1093/database/bau098.

Andersen, M. S., J. Dahl, and L. Vandenberghe (2003). *CVXOPT: A Python package for convex optimization*. cvxopt.org.

Assenov, Yassen, Fabian Müller, Pavlo Lutsik, Jörn Walter, Thomas Lengauer, and Christoph Bock (Nov. 2014). "Comprehensive Analysis of DNA Methylation Data with RnBeads". *Nature Methods* 11.11, 1138–1140. DOI: 10.1038/nmeth.3115.

Begemann, Matthias, Faisal I Rezwan, Jasmin Beygo, Louise E Docherty, Julia Kolarova, Christopher Schroeder, Karin Buiting, Kamal Chokkalingam, Franziska Degenhardt, Emma L Wakeling, Stephanie Kleinle, Daniela González Fassrainer, Barbara Oehl-Jaschkowitz, Claire L S Turner, Michal Patalan, Maria Gizewska, Gerhard Binder, Can Thi Bich Ngoc, Vu Chi Dung, Sarju G Mehta, Gareth Baynam, Julian P Hamilton-Shield, Sara Aljareh, Oluwakemi Lokulo-Sodipe, Rachel Horton, Reiner Siebert, Miriam Elbracht, Isabel Karen Temple, Thomas Eggermann, and Deborah J G Mackay (July 2018). "Maternal variants in NLRP and other maternal effect proteins are associated with multilocus imprinting disturbance in offspring". *Journal of Medical Genetics* 55.7, 497–504. DOI: 10.1136/jmedgenet-2017-105190.

Berdasco, María and Manel Esteller (2011). "DNA methylation in stem cell renewal and multipotency". *Stem Cell Research & Therapy* 2, 42–51. DOI: 10.1186/scrt83.

Bhasin, Jeffrey M, Bo Hu, and Angela H Ting (Jan. 2016). "MethylAction: detecting differentially methylated regions that distinguish biological subtypes." *Nucleic acids research* 44.1, 106–16. DOI: 10.1093/nar/gkv1461.

Bock, Christoph (Oct. 2012). "Analysing and interpreting DNA methylation data". *Nature Reviews Genetics* 13.10, 705–719. DOI: 10.1038/nrg3273.

Bresch, C. and R. Hausmann (1972). *Klassische und molekulare Genetik*. Berlin, Heidelberg: Springer Berlin Heidelberg. DOI: 10.1007/978-3-642-87168-9.

Broad Institute (2013). *Picard Tools*. http://broadinstitute.github.io/picard/.

Bujold, David, David Anderson de Lima Morais, Carol Gauthier, Catherine Côté, Maxime Caron, Tony Kwan, Kuang Chung Chen, Jonathan Laperle, Alexei Nordell Markovits, Tomi Pastinen, Bryan Caron, Alain Veilleux, Pierre-Étienne Jacques, and Guillaume Bourque (Nov. 2016).

"The International Human Epigenome Consortium Data Portal". *Cell Systems* 3.5, 496–499.e2. DOI: 10.1016/j.cels.2016.10.019.

Chen, Pao-Yang, Shawn J. Cokus, and Matteo Pellegrini (2010). "BS Seeker: precise mapping for bisulfite sequencing". *BMC Bioinformatics* 11, 203–209. DOI: 10.1186/1471-2105-11-203.

Cibulskis, Kristian, Michael S Lawrence, Scott L Carter, Andrey Sivachenko, David Jaffe, Carrie Sougnez, Stacey Gabriel, Matthew Meyerson, Eric S Lander, and Gad Getz (Mar. 2013). "Sensitive detection of somatic point mutations in impure and heterogeneous cancer samples". *Nature Biotechnology* 31.3, 213–219. DOI: 10.1038/nbt.2514.

Cingolani, P., A. Platts, M. Coon, T. Nguyen, L. Wang, S.J. Land, X. Lu, and D.M. Ruden (2012a). "A program for annotating and predicting the effects of single nucleotide polymorphisms, SnpEff: SNPs in the genome of Drosophila melanogaster strain w^{1118}; iso-2; iso-3". *Fly* 6.2, 80–92.

Cingolani, Pablo, Adrian Platts, Le Lily Wang, Melissa Coon, Tung Nguyen, Luan Wang, Susan J. Land, Xiangyi Lu, and Douglas M. Ruden (June 2012b). "A program for annotating and predicting the effects of single nucleotide polymorphisms, SnpEff: SNPs in the genome of Drosophila melanogaster strain w1118; iso-2; iso-3". *Fly* 6.2, 80–92. DOI: 10.4161/fly.19695.

Collette, Andrew (2008). *HDF5 for Python*. http://h5py.alfven.org.

Coutant, Sophie, Chloé Cabot, Arnaud Lefebvre, Martine Léonard, Elise Prieur-Gaston, Dominique Campion, Thierry Lecroq, and Hélène Dauchel (2012). "EVA: Exome Variation Analyzer, an efficient and versatile tool for filtering strategies in medical genomics". *BMC Bioinformatics* 13 Suppl 14, S9. DOI: 10.1186/1471-2105-13-S14-S9.

Cross, Sally H. and Adrian P. Bird (June 1995). "CpG islands and genes". *Current Opinion in Genetics & Development* 5.3, 309–314.

Daley, Timothy and Andrew D. Smith (Apr. 2013). "Predicting the molecular complexity of sequencing libraries". *Nature Methods* 10.4, 325–327. DOI: 10.1038/nmeth.2375.

Danecek, Petr, Adam Auton, Goncalo Abecasis, Cornelis A. Albers, Eric Banks, Mark A. DePristo, Robert E. Handsaker, Gerton Lunter, Gabor T. Marth, Stephen T. Sherry, Gilean McVean, and Richard Durbin (Aug. 2011). "The variant call format and VCFtools". *Bioinformatics* 27.15, 2156–2158. DOI: 10.1093/bioinformatics/btr330.

Dempster, A. P., N. M. Laird, and D. B. Rubin (1977). "Maximum likelihood from incomplete data via the EM algorithm". *Journal of the Royal Statistical Society, Series B* 39.1, 1–38.

Do, Catherine, Charles F. Lang, John Lin, Huferesh Darbary, Izabela Krupska, Aulona Gaba, Lynn Petukhova, Jean-Paul Vonsattel, Mary P. Gallagher, Robin S. Goland, Raphael A. Clynes, Andrew Dwork, John G. Kral, Catherine Monk, Angela M. Christiano, and Benjamin Tycko (May 2016). "Mechanisms and Disease Associations of Haplotype-Dependent Allele-Specific DNA Methylation". *American Journal of Human Genetics* 98.5, 934–955. DOI: 10.1016/j.ajhg.2016.03.027.

Du, Pan, Xiao Zhang, Chiang-Ching Huang, Nadereh Jafari, Warren A. Kibbe, Lifang Hou, and Simon M. Lin (Nov. 2010). "Comparison of Beta-value and M-value methods for quantifying methylation levels by microarray analysis". *BMC bioinformatics* 11, 587. DOI: 10.1186/1471-2105-11-587.

Dupont, Cathérine, D. Armant, and Carol Brenner (Sept. 2009). “Epigenetics: Definition, Mechanisms and Clinical Perspective”. *Seminars in Reproductive Medicine* 27.05, 351–357. DOI: 10.1055/s-0029-1237423.

Elsa, Leitão, Stefan Wallner, Christopher Schröder, Gerd Schmitz, and Bernhard Horsthemke (2014). *Epigenetic dynamics of monocyte to macrophage differentiation*. Gesellschaft für Humangenetik, Lübeck.

Erbel, R., L. Eisele, S. Moebus, N. Dragano, S. Möhlenkamp, M. Bauer, H. Kälsch, and K.-H. Jöckel (June 2012). “The Heinz Nixdorf Recall study”. *Bundesgesundheitsblatt, Gesundheitsforschung, Gesundheitsschutz* 55.6-7, 809–815. DOI: 10.1007/s00103-012-1490-7.

Ernst, Jason and Manolis Kellis (Feb. 2012). “ChromHMM: automating chromatin-state discovery and characterization”. *Nature Methods* 9.3, 215–216. DOI: 10.1038/nmeth.1906.

Ewing, B. and P. Green (Mar. 1998). “Base-calling of automated sequencer traces using phred. II. Error probabilities”. *Genome Research* 8.3, 186–194.

Ewing, B, L Hillier, M C Wendl, and P Green (Mar. 1998). “Base-calling of automated sequencer traces using phred. I. Accuracy assessment.” *Genome research* 8.3, 175–85. DOI: 10.1101/GR.8.3.175.

Fadista, João, Alisa K Manning, Jose C Florez, and Leif Groop (Aug. 2016). “The (in)famous GWAS P-value threshold revisited and updated for low-frequency variants”. *European Journal of Human Genetics* 24.8, 1202–1205. DOI: 10.1038/ejhg.2015.269.

Faust, Gregory G. and Ira M. Hall (Sept. 2014). “SAMBLASTER: fast duplicate marking and structural variant read extraction”. *Bioinformatics* 30.17, 2503–2505. DOI: 10.1093/bioinformatics/btu314.

Felsenfeld, Gary and Mark Groudine (Jan. 2003). “Controlling the double helix”. *Nature* 421.6921, 448–453. DOI: 10.1038/nature01411.

Fischer, Bernd and Gregoire Pau (2016). *rhdf5: HDF5 interface to R*.

Gardiner-Garden, M. and M. Frommer (July 1987). “CpG Islands in vertebrate genomes”. *Journal of Molecular Biology* 196.2, 261–282. DOI: 10.1016/0022-2836(87)90689-9.

Garrison, Erik and Gabor Marth (July 2012). “Haplotype-based variant detection from short-read sequencing”. *arXiv:1207.3907 [q-bio]*.

Genome Reference Consortium. https://www.ncbi.nlm.nih.gov/grc.

Gibbs, J. Raphael, Marcel P. van der Brug, Dena G. Hernandez, Bryan J. Traynor, Michael A. Nalls, Shiao-Lin Lai, Sampath Arepalli, Allissa Dillman, Ian P. Rafferty, Juan Troncoso, Robert Johnson, H. Ronald Zielke, Luigi Ferrucci, Dan L. Longo, Mark R. Cookson, and Andrew B. Singleton (May 2010). “Abundant quantitative trait loci exist for DNA methylation and gene expression in human brain”. *PLoS Genetics* 6.5, e1000952. DOI: 10.1371/journal.pgen.1000952.

Good, I. J. and G. H. Toulmin (1956). “The number of new species, and the increase in population coverage, when a sample is increased”. *Biometrika* 43.1–2, 45–63.

Granata, Ilaria, Mara Sangiovanni, Francesco Maiorano, Marco Miele, and Mario Rosario Guarracino (Nov. 2016). “Var2GO: a web-based tool for gene variants selection”. *BMC Bioinformatics* 17.Suppl 12, 376. DOI: 10.1186/s12859-016-1197-0.

Grün, Bettina, Ioannis Kosmidis, and Achim Zeileis (2012). “Extended Beta Regression in R: Shaken, Stirred, Mixed, and Partitioned”. *Journal of Statistical Software* 48.11, 1–25.

Grüning, Björn, Ryan Dale, Andreas Sjödin, Jillian Rowe, Brad A. Chapman, Christopher H. Tomkins-Tinch, Renan Valieris, The Bioconda Team, and Johannes Köster (Oct. 2017). "Bioconda: A sustainable and comprehensive software distribution for the life sciences". *bioRxiv*, 207092. DOI: 10.1101/207092.

Guttman, Antonin (1984). "R-trees: A Dynamic Index Structure for Spatial Searching". *Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data*. SIGMOD '84. New York, NY, USA: ACM, 47–57. DOI: 10.1145/602259.602266.

Hackett, Jamie A and M Azim Surani (2013). "DNA methylation dynamics during the mammalian life cycle". *Philosophical Transactions of the Royal Society of London B: Biological Sciences* 368.1609. DOI: 10.1098/rstb.2011.0328.

Hagberg, Aric A., Daniel A. Schult, and Pieter J. Swart (Aug. 2008). "Exploring network structure, dynamics, and function using NetworkX". *Proceedings of the 7th Python in Science Conference (SciPy2008)*. Pasadena, CA USA, 11–15.

Hajian-Tilaki, Karimollah (2013). "Receiver Operating Characteristic (ROC) Curve Analysis for Medical Diagnostic Test Evaluation". *Caspian Journal of Internal Medicine* 4.2, 627–635.

Hansen, Kasper D., Benjamin Langmead, and Rafael A. Irizarry (2012). "BSmooth: from whole genome bisulfite sequencing reads to differentially methylated regions". *Genome Biology* 13.10, R83. DOI: 10.1186/gb-2012-13-10-r83.

Harris, Elena Y., Nadia Ponts, Le Roch, Karine G, and Stefano Lonardi (July 2012). "BRAT-BW: efficient and accurate mapping of bisulfite-treated reads". *Bioinformatics* 28.13, 1795–1796. DOI: 10.1093/bioinformatics/bts264.

Hesse, Nina, Christopher Schröder, and Sven Rahmann (2014). *An optimization approach to detect differentially methylated regions from Whole Genome Bisulfite Sequencing data*. German Conference on Bioinformatics, Dortmund.

Heyn, Holger, Ning Li, Humberto J. Ferreira, Sebastian Moran, David G. Pisano, Antonio Gomez, Javier Diez, Jose V. Sanchez-Mut, Fernando Setien, F. Javier Carmona, Annibale A. Puca, Sergi Sayols, Miguel A. Pujana, Jordi Serra-Musach, Isabel Iglesias-Platas, Francesc Formiga, Agustin F. Fernandez, Mario F. Fraga, Simon C. Heath, Alfonso Valencia, Ivo G. Gut, Jun Wang, and Manel Esteller (June 2012). "Distinct DNA methylomes of newborns and centenarians". *Proceedings of the National Academy of Sciences of the United States of America* 109.26, 10522–10527. DOI: 10.1073/pnas.1120658109.

Holy, Tim (2016). *HDF5.jl - Saving and loading data in the HDF5 file format*. https://github.com/JuliaLang/HDF5.jl.

Howie, Bryan N., Peter Donnelly, and Jonathan Marchini (June 2009). "A Flexible and Accurate Genotype Imputation Method for the Next Generation of Genome-Wide Association Studies". *PLoS Genetics* 5.6, e1000529. DOI: 10.1371/journal.pgen.1000529.

Hunter, J. D. (2007). "Matplotlib: A 2D graphics environment". *Computing In Science & Engineering* 9.3, 90–95. DOI: 10.1109/MCSE.2007.55.

Illingworth, Robert S., Ulrike Gruenewald-Schneider, Dina De Sousa, Shaun Webb, Cara Merusi, Alastair R. W. Kerr, Keith D. James, Colin

Smith, Robert Walker, Robert Andrews, and Adrian P. Bird (Jan. 2015). "Inter-individual variability contrasts with regional homogeneity in the human brain DNA methylome". *Nucleic Acids Research* 43.2, 732–744. DOI: 10.1093/nar/gku1305.

Illumina Inc (2017). *Sequencing by synthesis*. https://www.illumina.com/science/technology/next-generation-sequencing/sequencing-technology.html.

International HapMap Consortium (Dec. 2003). "The International HapMap Project". *Nature* 426.6968, 789–796. DOI: 10.1038/nature02168.

Ito-Harashima, Sayoko, Kazushige Kuroha, Tsuyako Tatematsu, and Toshifumi Inada (Mar. 2007). "Translation of the poly(A) tail plays crucial roles in nonstop mRNA surveillance via translation repression and protein destabilization by proteasome in yeast." *Genes & development* 21.5, 519–24. DOI: 10.1101/gad.1490207.

Jaffe, Andrew E, Peter Murakami, Hwajin Lee, Jeffrey T Leek, M Daniele Fallin, Andrew P Feinberg, and Rafael A Irizarry (Feb. 2012). "Bump hunting to identify differentially methylated regions in epigenetic epidemiology studies". *International Journal of Epidemiology* 41.1, 200–209. DOI: 10.1093/ije/dyr238.

Janetzki, Steffen, Magnús Rafn Tiedemann, and Hardik Balar (July 2015). *Genome Data Management using RDBMSs*. Tech. rep. University of Magdeburg. DOI: 10.13140/RG.2.1.4047.6006.

Johnson, Adiv A., Kemal Akman, Stuart R.G. Calimport, Daniel Wuttke, Alexandra Stolzing, and João Pedro de Magalhães (Oct. 2012). "The Role of DNA Methylation in Aging, Rejuvenation, and Age-Related Disease". *Rejuvenation Research* 15.5, 483–494. DOI: 10.1089/rej.2012.1324.

Jühling, Frank, Helene Kretzmer, Stephan H. Bernhart, Christian Otto, Peter F. Stadler, and Steve Hoffmann (Dec. 2015). "metilene: Fast and sensitive calling of differentially methylated regions from bisulfite sequencing data". *Genome Research*, 256–262. DOI: 10.1101/gr.196394.115.

Kaplinsky, Joseph and Ramy Arnaout (June 2016). "Robust estimates of overall immune-repertoire diversity from high-throughput measurements on samples". *Nature Communications* 7, 11881. DOI: 10.1038/ncomms11881.

Karczewski, Konrad J., Ben Weisburd, Brett Thomas, Matthew Solomonson, Douglas M. Ruderfer, David Kavanagh, Tymor Hamamsy, Monkol Lek, Kaitlin E. Samocha, Beryl B. Cummings, Daniel Birnbaum, Mark J. Daly, and Daniel G. MacArthur (Jan. 2017). "The ExAC browser: displaying reference data information from over 60 000 exomes". *Nucleic Acids Research* 45.Database issue, D840–D845. DOI: 10.1093/nar/gkw971.

Kelly, Theresa K., Yaping Liu, Fides D. Lay, Gangning Liang, Benjamin P. Berman, and Peter A. Jones (Dec. 2012). "Genome-wide mapping of nucleosome positioning and DNA methylation within individual DNA molecules". *Genome Research* 22.12, 2497–2506. DOI: 10.1101/gr.143008.112.

Kent, W. James, Charles W. Sugnet, Terrence S. Furey, Krishna M. Roskin, Tom H. Pringle, Alan M. Zahler, and and David Haussler (June 2002). "The Human Genome Browser at UCSC". *Genome Research* 12.6, 996–1006. DOI: 10.1101/gr.229102.

Krueger, Felix and Simon R. Andrews (June 2011). “Bismark: a flexible aligner and methylation caller for Bisulfite-Seq applications”. *Bioinformatics (Oxford, England)* 27.11, 1571–1572. DOI: 10.1093/bioinformatics/btr167.

Köster, Johannes and Sven Rahmann (Oct. 2012). “Snakemake–a scalable bioinformatics workflow engine”. *Bioinformatics (Oxford, England)* 28.19, 2520–2522. DOI: 10.1093/bioinformatics/bts480.

Lanchantin, Jack, Ritambhara Singh, Zeming Lin, and Yanjun Qi (May 2016). “Deep Motif: Visualizing Genomic Sequence Classifications”. arXiv: 1605.01133.

Landt, Stephen G., Georgi K. Marinov, Anshul Kundaje, Pouya Kheradpour, Florencia Pauli, Serafim Batzoglou, Bradley E. Bernstein, Peter Bickel, James B. Brown, Philip Cayting, Yiwen Chen, Gilberto DeSalvo, Charles Epstein, Katherine I. Fisher-Aylor, Ghia Euskirchen, Mark Gerstein, Jason Gertz, Alexander J. Hartemink, Michael M. Hoffman, Vishwanath R. Iyer, Youngsook L. Jung, Subhradip Karmakar, Manolis Kellis, Peter V. Kharchenko, Qunhua Li, Tao Liu, X. Shirley Liu, Lijia Ma, Aleksandar Milosavljevic, Richard M. Myers, Peter J. Park, Michael J. Pazin, Marc D. Perry, Debasish Raha, Timothy E. Reddy, Joel Rozowsky, Noam Shoresh, Arend Sidow, Matthew Slattery, John A. Stamatoyannopoulos, Michael Y. Tolstorukov, Kevin P. White, Simon Xi, Peggy J. Farnham, Jason D. Lieb, Barbara J. Wold, and Michael Snyder (Sept. 2012). “ChIP-seq guidelines and practices of the ENCODE and modENCODE consortia”. *Genome Research* 22.9, 1813–1831. DOI: 10.1101/gr.136184.111.

Langmead, Ben, Cole Trapnell, Mihai Pop, and Steven L. Salzberg (2009). “Ultrafast and memory-efficient alignment of short DNA sequences to the human genome”. *Genome Biology* 10, 25–35. DOI: 10.1186/gb-2009-10-3-r25.

Lek, Monkol, Konrad J. Karczewski, Eric V. Minikel, Kaitlin E. Samocha, Eric Banks, Timothy Fennell, Anne H. O’Donnell-Luria, James S. Ware, Andrew J. Hill, Beryl B. Cummings, Taru Tukiainen, Daniel P. Birnbaum, Jack A. Kosmicki, Laramie E. Duncan, Karol Estrada, Fengmei Zhao, James Zou, Emma Pierce-Hoffman, Joanne Berghout, David N. Cooper, Nicole Deflaux, Mark DePristo, Ron Do, Jason Flannick, Menachem Fromer, Laura Gauthier, Jackie Goldstein, Namrata Gupta, Daniel Howrigan, Adam Kiezun, Mitja I. Kurki, Ami Levy Moonshine, Pradeep Natarajan, Lorena Orozco, Gina M. Peloso, Ryan Poplin, Manuel A. Rivas, Valentin Ruano-Rubio, Samuel A. Rose, Douglas M. Ruderfer, Khalid Shakir, Peter D. Stenson, Christine Stevens, Brett P. Thomas, Grace Tiao, Maria T. Tusie-Luna, Ben Weisburd, Hong-Hee Won, Dongmei Yu, David M. Altshuler, Diego Ardissino, Michael Boehnke, John Danesh, Stacey Donnelly, Roberto Elosua, Jose C. Florez, Stacey B. Gabriel, Gad Getz, Stephen J. Glatt, Christina M. Hultman, Sekar Kathiresan, Markku Laakso, Steven McCarroll, Mark I. McCarthy, Dermot McGovern, Ruth McPherson, Benjamin M. Neale, Aarno Palotie, Shaun M. Purcell, Danish Saleheen, Jeremiah M. Scharf, Pamela Sklar, Patrick F. Sullivan, Jaakko Tuomilehto, Ming T. Tsuang, Hugh C. Watkins, James G. Wilson, Mark J. Daly, Daniel G. MacArthur, and Exome Aggregation Consortium (Aug. 2016). "Analysis of protein-coding genetic variation in 60,706 humans". *Nature* 536.7616, 285–291. DOI: 10.1038/nature19057.

Levene, P. A. (Dec. 1919). "The Structure of Yeast Nucleic Acid Iv. Ammonia Hydrolysis". *Journal of Biological Chemistry* 40.2, 415–424.

Li, H. and R. Durbin (July 2009). "Fast and accurate short read alignment with Burrows-Wheeler transform". *Bioinformatics* 25.14, 1754–1760. DOI: 10.1093/bioinformatics/btp324.

Li, Heng (Mar. 2013). "Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM". *arXiv:1303.3997 [q-bio]*. arXiv: 1303.3997.

Li, Heng, Jue Ruan, and Richard Durbin (Nov. 2008). "Mapping short DNA sequencing reads and calling variants using mapping quality scores". *Genome Research* 18.11, 1851–1858. DOI: 10.1101/gr.078212.108.

Li, Heng, Bob Handsaker, Alec Wysoker, Tim Fennell, Jue Ruan, Nils Homer, Gabor Marth, Goncalo Abecasis, Richard Durbin, and 1000 Genome Project Data Processing Subgroup (Aug. 2009). "The Sequence Alignment/Map format and SAMtools". *Bioinformatics (Oxford, England)* 25.16, 2078–2079. DOI: 10.1093/bioinformatics/btp352.

Lim, Jing-Quan, Chandana Tennakoon, Guoliang Li, Eleanor Wong, Yijun Ruan, Chia-Lin Wei, and Wing-Kin Sung (Oct. 2012). "BatMeth: improved mapper for bisulfite sequencing reads on DNA methylation". *Genome Biology* 13.10, R82. DOI: 10.1186/gb-2012-13-10-r82.

Liu, Y., F. R. Abdul Razak, M. Terpstra, F. C. Chan, A. Saber, M. Nijland, G. van Imhoff, L. Visser, R. Gascoyne, C. Steidl, J. Kluiver, A. Diepstra, K. Kok, and A. van den Berg (Nov. 2014). "The mutational landscape of Hodgkin lymphoma cell lines determined by whole-exome sequencing". *Leukemia* 28.11, 2248–2251.

Liu, Yaping, Kimberly D Siegmund, Peter W Laird, and Benjamin P Berman (2012). "Bis-SNP: Combined DNA methylation and SNP calling for Bisulfite-seq data". *Genome Biology* 13.7, R61. DOI: 10.1186/gb-2012-13-7-r61.

Maaten, Laurens van der and Geoffrey Hinton (2008). "Visualizing Data using t-SNE". *Journal of Machine Learning Research* 9, 2579–2605.

Maksimovic, Jovana, Lavinia Gordon, and Alicia Oshlack (June 2012). "SWAN: Subset-quantile Within Array Normalization for Illumina Infinium HumanMethylation450 BeadChips". *Genome Biology* 13.6, R44. DOI: 10.1186/gb-2012-13-6-r44.

Martin, Marcel (2013). "Algorithms and Tools for the Analysis of High-Throughput DNA Sequencing Data". PhD thesis. Dortmund, Germany: TU Dortmund.

Martin, Marcel, Lars Maßhöfer, Petra Temming, Sven Rahmann, Claudia Metz, Norbert Bornfeld, Johannes van de Nes, Ludger Klein-Hitpass, Alan G Hinnebusch, Bernhard Horsthemke, Dietmar R Lohmann, and Michael Zeschnigk (Aug. 2013). "Exome sequencing identifies recurrent somatic mutations in EIF1AX and SF3B1 in uveal melanoma with disomy 3". *Nature genetics* 45.8, 933–936. DOI: 10.1038/ng.2674.

Masetti, R., I. Castelli, A. Astolfi, S. N. Bertuccio, V. Indio, M. Togni, T. Belotti, S. Serravalle, G. Tarantino, M. Zecca, M. Pigazzi, G. Basso, A. Pession, and F. Locatelli (Aug. 2016). "Genomic complexity and dynamics of clonal evolution in childhood acute myeloid leukemia studied with whole-exome sequencing". *Oncotarget* 7.35, 56746–56757.

Matys, V., O. V. Kel-Margoulis, E. Fricke, I. Liebich, S. Land, A. Barre-Dirrie, I. Reuter, D. Chekmenev, M. Krull, K. Hornischer, N. Voss, P. Stegmaier, B. Lewicki-Potapov, H. Saxel, A. E. Kel, and E. Wingender (Jan. 2006). "TRANSFAC and its module TRANSCompel: transcriptional gene regulation in eukaryotes". *Nucleic Acids Research* 34.Database issue, D108–110. DOI: 10.1093/nar/gkj143.

Maxmen, A. (Mar. 2011). "Exome sequencing deciphers rare diseases". *Cell* 144.5, 635–637.

McDonald, John H. (2014). *Handbook of Biological Statistics (3rd ed.)* Third Edition. Baltimore, Maryland, USA: Sparky House Publishing, 293–296.

McKenna, Aaron, Matthew Hanna, Eric Banks, Andrey Sivachenko, Kristian Cibulskis, Andrew Kernytsky, Kiran Garimella, David Altshuler, Stacey Gabriel, Mark Daly, and Mark A. DePristo (Sept. 2010). "The Genome Analysis Toolkit: A MapReduce framework for analyzing next-generation DNA sequencing data". *Genome Research* 20.9, 1297–1303. DOI: 10.1101/gr.107524.110.

McKusick, Victor A. (Apr. 2007). "Mendelian Inheritance in Man and Its Online Version, OMIM". *American Journal of Human Genetics* 80.4, 588–604.

McLean, Cory Y., Dave Bristor, Michael Hiller, Shoa L. Clarke, Bruce T. Schaar, Craig B. Lowe, Aaron M. Wenger, and Gill Bejerano (May 2010). "GREAT improves functional interpretation of cis-regulatory regions". *Nature Biotechnology* 28.5, 495–501. DOI: 10.1038/nbt.1630.

Muir, Paul, Shantao Li, Shaoke Lou, Daifeng Wang, Daniel J. Spakowicz, Leonidas Salichos, Jing Zhang, George M. Weinstock, Farren Isaacs, Joel Rozowsky, and Mark Gerstein (Mar. 2016). "The real cost of sequencing: scaling computation to keep pace with data generation". *Genome Biology* 17, 53–62. DOI: 10.1186/s13059-016-0917-0.

Nambiar, Mridula and Sathees C. Raghavan (Aug. 2011). "How does DNA break during chromosomal translocations?" *Nucleic Acids Research* 39.14, 5813–5825. DOI: 10.1093/nar/gkr223.

Nguyen, M. T. and K. Charlebois (Oct. 2015). "The clinical utility of whole-exome sequencing in the context of rare diseases - the changing tides of medical practice". *Clin. Genet.* 88.4, 313–319.

Nielsen, Rasmus, Joshua S. Paul, Anders Albrechtsen, and Yun S. Song (June 2011). "Genotype and SNP calling from next-generation sequencing data". *Nature reviews. Genetics* 12.6, 443–451. DOI: 10.1038/nrg2986.

Novus Partners (2014). *NVD3*. http://nvd3.org/.

O'Leary, Nuala A., Mathew W. Wright, J. Rodney Brister, Stacy Ciufo, Diana Haddad, Rich McVeigh, Bhanu Rajput, Barbara Robbertse, Brian Smith-White, Danso Ako-Adjei, Alexander Astashyn, Azat Badretdin, Yiming Bao, Olga Blinkova, Vyacheslav Brover, Vyacheslav Chetvernin, Jinna Choi, Eric Cox, Olga Ermolaeva, Catherine M. Farrell, Tamara Goldfarb, Tripti Gupta, Daniel Haft, Eneida Hatcher, Wratko Hlavina, Vinita S. Joardar, Vamsi K. Kodali, Wenjun Li, Donna Maglott, Patrick Masterson, Kelly M. McGarvey, Michael R. Murphy, Kathleen O'Neill, Shashikant Pujar, Sanjida H. Rangwala, Daniel Rausch, Lillian D. Riddick, Conrad Schoch, Andrei Shkeda, Susan S. Storz, Hanzhen Sun, Francoise Thibaud-Nissen, Igor Tolstoy, Raymond E. Tully, Anjana R. Vatsan, Craig Wallin, David Webb, Wendy Wu, Melissa J. Landrum, Avi Kimchi, Tatiana Tatusova, Michael DiCuccio, Paul Kitts, Terence D. Murphy, and Kim D. Pruitt (Jan. 2016). "Reference sequence (RefSeq) database at NCBI: current status, taxonomic expansion, and functional annotation". *Nucleic Acids Research* 44.D1, D733–D745. DOI: 10.1093/nar/gkv1189.

Paila, Umadevi, Brad A. Chapman, Rory Kirchner, and Aaron R. Quinlan (July 2013). "GEMINI: Integrative Exploration of Genetic Variation and Genome Annotations". *PLOS Computational Biology* 9.7, e1003153. DOI: 10.1371/journal.pcbi.1003153.

Paliwal, Anupam, Alexis M. Temkin, Kristi Kerkel, Alexander Yale, Iveta Yotova, Natalia Drost, Simon Lax, Chia-Ling Nhan-Chang, Charles Powell, Alain Borczuk, Abraham Aviv, Ronald Wapner, Xiaowei Chen, Peter L. Nagy, Nicholas Schork, Catherine Do, Ali Torkamani, and Benjamin Tycko (Aug. 2013). "Comparative anatomy of chromosomal domains with imprinted and non-imprinted allele-specific DNA methylation". *PLoS Genetics* 9.8, e1003622. DOI: 10.1371/journal.pgen.1003622.

Pavlopoulos, Georgios A., Anastasis Oulas, Ernesto Iacucci, Alejandro Sifrim, Yves Moreau, Reinhard Schneider, Jan Aerts, and Ioannis Iliopoulos (July 2013). "Unraveling genomic variation from next generation sequencing data". *BioData Mining* 6, 13–38. DOI: 10.1186/1756-0381-6-13.

Pedersen, Brent, Tzung-Fu Hsieh, Christian Ibarra, and Robert L. Fischer (Sept. 2011). "MethylCoder: software pipeline for bisulfite-treated sequences". *Bioinformatics* 27.17, 2435–2436. DOI: 10.1093/bioinformatics/btr394.

Pedersen, Brent S., Kenneth Eyring, Subhajyoti De, Ivana V. Yang, and David A. Schwartz (Jan. 2014). "Fast and accurate alignment of long bisulfite-seq reads". *arXiv:1401.1129 [q-bio]*.

Pitman, Jim (June 1995). "Exchangeable and partially exchangeable random partitions". *Probability Theory and Related Fields* 102.2, 145–158. DOI: 10.1007/BF01213386.

Poplin, Ryan, Dan Newburger, Jojo Dijamco, Nam Nguyen, Dion Loy, Sam S. Gross, Cory Y. McLean, and Mark A. DePristo (Jan. 2018). "Creating a universal SNP and small indel variant caller with deep neural networks". *bioRxiv*, 092890. DOI: 10.1101/092890.

Pounds, Stan and Stephan W. Morris (July 2003). "Estimating the occurrence of false positives and false negatives in microarray studies by approximating and partitioning the empirical distribution of p-values". *Bioinformatics* 19.10, 1236–1242. DOI: 10.1093/bioinformatics/btg148.

Purcell, Shaun, Benjamin Neale, Kathe Todd-Brown, Lori Thomas, Manuel A. R. Ferreira, David Bender, Julian Maller, Pamela Sklar, Paul I. W. de Bakker, Mark J. Daly, and Pak C. Sham (Sept. 2007). "PLINK: a tool set for whole-genome association and population-based linkage analyses". *American Journal of Human Genetics* 81.3, 559–575. DOI: 10.1086/519795.

Rademacher, Katrin, Christopher Schröder, Deniz Kanber, Ludger Klein-Hitpass, Stefan Wallner, Michael Zeschnigk, and Bernhard Horsthemke

(July 2014). “Evolutionary Origin and Methylation Status of Human Intronic CpG Islands that Are Not Present in Mouse”. *Genome Biology and Evolution* 6.7, 1579–1588. DOI: 10.1093/gbe/evu125.

Ramensky, Vasily, Peer Bork, and Shamil Sunyaev (Sept. 2002). “Human non-synonymous SNPs: server and survey”. *Nucleic Acids Research* 30.17, 3894–3900.

Rausch, Tobias, Thomas Zichner, Andreas Schlattl, Adrian M. Stütz, Vladimir Benes, and Jan O. Korbel (Sept. 2012). “DELLY: structural variant discovery by integrated paired-end and split-read analysis”. *Bioinformatics (Oxford, England)* 28.18, i333–i339. DOI: 10.1093/bioinformatics/bts378.

Redner, R. A. and H. F. Walker (1984). “Mixture densities, maximum likelihood, and the EM algorithm”. *SIAM Review* 26, 195–239.

Roach, J. C., G. Glusman, A. F. A. Smit, C. D. Huff, R. Hubley, P. T. Shannon, L. Rowen, K. P. Pant, N. Goodman, M. Bamshad, J. Shendure, R. Drmanac, L. B. Jorde, L. Hood, and D. J. Galas (Apr. 2010). “Analysis of Genetic Inheritance in a Family Quartet by Whole-Genome Sequencing”. *Science* 328.5978, 636–639. DOI: 10.1126/science.1186802.

Robertson, Keith D. (Aug. 2005). “DNA methylation and human disease”. *Nature Reviews Genetics* 6.8, 597–610. DOI: 10.1038/nrg1655.

Robinson, James T., Helga Thorvaldsdóttir, Wendy Winckler, Mitchell Guttman, Eric S. Lander, Gad Getz, and Jill P. Mesirov (Jan. 2011). “Integrative genomics viewer”. *Nature Biotechnology* 29.1, 24–26. DOI: 10.1038/nbt.1754.

Ronacher, Armin (2008). *Jinja2*. jinja.pocoo.org/.

— (2017). *Flask (a Python microframework)*. http://flask.pocoo.org/.

Rubio-Camarillo, Miriam, Gonzalo Gómez-López, José M. Fernández, Alfonso Valencia, and David G. Pisano (2013). “RUbioSeq: A suite of parallelized pipelines to automate exome variation and bisulfite-seq analyses”. *Bioinformatics* 29.13, 1687–1689. DOI: 10.1093/bioinformatics/btt203.

Saito, Yutaka, Junko Tsuji, and Toutai Mituyama (Apr. 2014). “Bisulfighter: accurate detection of methylated cytosines and differentially methylated regions”. *Nucleic Acids Research* 42.6, e45–e45. DOI: 10.1093/nar/gkt1373.

Sanger, F. and A. R. Coulson (May 1975). “A rapid method for determining sequences in DNA by primed synthesis with DNA polymerase”. *Journal of Molecular Biology* 94.3, 441–448. DOI: 10.1016/0022-2836(75)90213-2.

Sanger, F., S. Nicklen, and A. R. Coulson (Dec. 1977). “DNA sequencing with chain-terminating inhibitors”. *Proceedings of the National Academy of Sciences of the United States of America* 74.12, 5463–5467.

Sauna, Zuben E. and Chava Kimchi-Sarfaty (Aug. 2011). “Understanding the contribution of synonymous mutations to human disease”. *Nature Reviews. Genetics* 12.10, 683–691. DOI: 10.1038/nrg3051.

Savitzky, Abraham and Marcel J. E. Golay (July 1964). “Smoothing and Differentiation of Data by Simplified Least Squares Procedures.” *Analytical Chemistry* 36.8, 1627–1639. DOI: 10.1021/ac60214a047.

Schalkwyk, Leonard C., Emma L. Meaburn, Rebecca Smith, Emma L. Dempster, Aaron R. Jeffries, Matthew N. Davies, Robert Plomin, and Jonathan Mill (Feb. 2010). “Allelic skewing of DNA methylation is widespread across the genome”. *American Journal of Human Genetics* 86.2, 196–212. DOI: 10.1016/j.ajhg.2010.01.014.

Schaller, Robert R. (June 1997). “Moore’s Law: Past, Present, and Future”. *IEEE Spectr.* 34.6, 52–59. DOI: 10.1109/6.591665.

Schramm, Alexander, Johannes Köster, Yassen Assenov, Kristina Althoff, Martin Peifer, Ellen Mahlow, Andrea Odersky, Daniela Beisser, Corinna Ernst, Anton G. Henssen, Harald Stephan, Christopher Schröder, Lukas Heukamp, Anne Engesser, Yvonne Kahlert, Jessica Theissen, Barbara Hero, Frederik Roels, Janine Altmüller, Peter Nürnberg, Kathy Astrahantseff, Christian Gloeckner, Katleen De Preter, Christoph Plass, Sangkyun Lee, Holger N. Lode, Kai-Oliver Henrich, Moritz Gartlgruber, Frank Speleman, Peter Schmezer, Frank Westermann, Sven Rahmann, Matthias Fischer, Angelika Eggert, and Johannes H. Schulte (Aug. 2015). “Mutational dynamics between primary and relapse neuroblastomas”. *Nature Genetics* 47.8, 872–877. DOI: 10.1038/ng.3349.

Schröder, Christopher and Sven Rahmann (2015). “Efficient duplicate rate estimation from subsamples of sequencing libraries”. *PeerJ PrePrints*. DOI: 10.7287/peerj.preprints.1298v2.

Schröder, Christopher and Sven Rahmann (2014). *Efficient duplicate rate estimation from subsamples of sequencing libraries*. German Conference on Bioinformatics, Dortmund.

— (Aug. 2017). “A hybrid parameter estimation algorithm for beta mixtures and applications to methylation state classification”. *Algorithms for Molecular Biology* 12, 21–33. DOI: 10.1186/s13015-017-0112-1.

Schröder, Christopher, Johannes Köster, and Sven Rahmann (2016a). *Interactivity vs. Reproducibility in High-Throughput Data Analysis*. German Bioinformatics Core Units Workshop, BIH, GCB Satellite, Berlin.

Schröder, Christopher, Nina Hesse, Johannes Köster, and Sven Rahmann (2016b). *Methods for discovering differentially methylated regions from whole genome bisulfite sequencing data (and keeping track of the process)*. GMDS Workshop: From Biomedical Informatics to Medical Bioinformatics and back HEC, München.

Schröder, Christopher, Elsa Leitão, Stefan Wallner, Gerd Schmitz, Ludger Klein-Hitpass, Anupam Sinha, Karl-Heinz Jöckel, Stefanie Heilmann-Heimbach, Per Hoffmann, Markus M. Nöthen, Michael Steffens, Peter Ebert, Sven Rahmann, and Bernhard Horsthemke (July 2017). “Regions of common inter-individual DNA methylation differences in human monocytes: genetic basis and potential function”. *Epigenetics & Chromatin* 10, 37–55. DOI: 10.1186/s13072-017-0144-2.

Schwarz, Jana Marie, David N. Cooper, Markus Schuelke, and Dominik Seelow (Apr. 2014). “MutationTaster2: mutation prediction for the deep-sequencing age”. *Nature Methods* 11.4, 361–362. DOI: 10.1038/nmeth.2890.

Sherry, S. T., M.-H. Ward, M. Kholodov, J. Baker, L. Phan, E. M. Smigielski, and K. Sirotkin (Jan. 2001). “dbSNP: the NCBI database of genetic variation”. *Nucleic Acids Research* 29.1, 308–311.

Singh, Ritambhara, Jack Lanchantin, Gabriel Robins, and Yanjun Qi (July 2016). “DeepChrome: Deep-learning for predicting gene expression from histone modifications”. arXiv: 1607.02078.

Slatkin, Montgomery (June 2008). “Linkage disequilibrium — understanding the evolutionary past and mapping the medical future”. *Nature reviews. Genetics* 9.6, 477–485. DOI: 10.1038/nrg2361.

Smith, Andrew D., Wen-Yu Chung, Emily Hodges, Jude Kendall, Greg Hannon, James Hicks, Zhenyu Xuan, and Michael Q. Zhang (Nov. 2009). “Updates to the RMAP short-read mapping software”. *Bioinformatics* 25.21, 2841–2842. DOI: 10.1093/bioinformatics/btp533.

Stunnenberg, Hendrik G. et al. (Nov. 2016). “The International Human Epigenome Consortium: A Blueprint for Scientific Collaboration and Discovery”. *Cell* 167.5, 1145–1149. DOI: 10.1016/j.cell.2016.11.007.

Su, Jianzhong, Haidan Yan, Yanjun Wei, Hongbo Liu, Hui Liu, Fang Wang, Jie Lv, Qiong Wu, and Yan Zhang (Jan. 2013). “CpG_MPs: identification of CpG methylation patterns of genomic regions from high-throughput bisulfite sequencing data”. *Nucleic Acids Research* 41.1, e4–e4. DOI: 10.1093/nar/gks829.

Suzuki, Kenji (Sept. 2017). “Overview of deep learning in medical imaging”. *Radiological Physics and Technology* 10.3, 257–273. DOI: 10.1007/s12194-017-0406-5.

Suzuki, Miho M. and Adrian Bird (June 2008). “DNA methylation landscapes: provocative insights from epigenomics”. *Nature Reviews. Genetics* 9.6, 465–476. DOI: 10.1038/nrg2341.

Tauber, Stefanie and Arndt von Haeseler (May 2013). “Exploring the sampling universe of RNA-seq”. *Statistical Applications in Genetics and Molecular Biology* 12.2, 175–188. DOI: 10.1515/sagmb-2012-0049.

Teer, Jamie K., Eric D. Green, James C. Mullikin, and Leslie G. Biesecker (Feb. 2012). “VarSifter: Visualizing and analyzing exome-scale sequence variation data on a desktop computer”. *Bioinformatics* 28.4, 599–600. DOI: 10.1093/bioinformatics/btr711.

The 1000 Genomes Project Consortium (Oct. 2015). “A global reference for human genetic variation”. *Nature* 526.7571, 68–74. DOI: 10.1038/nature15393.

The HDF Group (1997-2018). *Hierarchical Data Format, version 5*. http://www.hdfgroup.org/HDF5/.

Thorvaldsdóttir, Helga, James T. Robinson, and Jill P. Mesirov (Mar. 2013). “Integrative Genomics Viewer (IGV): high-performance genomics data visualization and exploration”. *Briefings in Bioinformatics* 14.2, 178–192. DOI: 10.1093/bib/bbs017.

Vazirani, Vijay V. (2001). *Approximation algorithms*. Springer.

Wallner, Stefan, Christopher Schröder, Elsa Leitão, Tea Berulava, Claudia Haak, Daniela Beißer, Sven Rahmann, Andreas S. Richter, Thomas Manke, Ulrike Bönisch, Laura Arrigoni, Sebastian Fröhler, Filippos Klironomos, Wei Chen, Nikolaus Rajewsky, Fabian Müller, Peter Ebert, Thomas Lengauer, Matthias Barann, Philip Rosenstiel, Gilles Gasparoni, Karl Nordström, Jörn Walter, Benedikt Brors, Gideon Zipprich, Bärbel Felder, Ludger Klein-Hitpass, Corinna Attenberger, Gerd Schmitz, and Bernhard Horsthemke (2016). “Epigenetic dynamics of monocyte-to-macrophage differentiation”. *Epigenetics & Chromatin* 9, 33–50. DOI: 10.1186/s13072-016-0079-z.

Walt, Stéfan van der, S. Chris Colbert, and Gaël Varoquaux (Mar. 2011). “The NumPy Array: A Structure for Efficient Numerical Computation”.

Computing in Science & Engineering 13.2, 22–30. DOI: 10.1109/MCSE.2011.37.

Wang, Zhen, Xianfeng Li, Yi Jiang, Qianzhi Shao, Qi Liu, BingYu Chen, and Dongsheng Huang (July 2015). "swDMR: A Sliding Window Approach to Identify Differentially Methylated Regions Based on Whole Genome Bisulfite Sequencing". *PLoS One* 10.7, e0132866. DOI: 10.1371/journal.pone.0132866.

Ward, Lucas D. and Manolis Kellis (Jan. 2012). "HaploReg: a resource for exploring chromatin states, conservation, and regulatory motif alterations within sets of genetically linked variants". *Nucleic Acids Research* 40.Database issue, D930–D934. DOI: 10.1093/nar/gkr917.

Watson, James D. and Francis. H. Crick (Apr. 1953). "Molecular structure of nucleic acids; a structure for deoxyribose nucleic acid". *Nature* 171.4356, 737–738.

Wen, Yalu, Fushun Chen, Qingzheng Zhang, Yan Zhuang, and Zhiguang Li (Aug. 2016). "Detection of differentially methylated regions in whole genome bisulfite sequencing data using local Getis-Ord statistics". *Bioinformatics* 32.22, btw497. DOI: 10.1093/bioinformatics/btw497.

Wieczorek, Dagmar, Nina Bögershausen, Filippo Beleggia, Sabine Steiner-Haldenstätt, Esther Pohl, Yun Li, Esther Milz, Marcel Martin, Holger Thiele, Janine Altmüller, Yasemin Alanay, Hülya Kayserili, Ludger Klein-Hitpass, Stefan Böhringer, Andreas Wollstein, Beate Albrecht, Koray Boduroglu, Almuth Caliebe, Krystyna Chrzanowska, Ozgur Cogulu, Francesca Cristofoli, Johanna Christina Czeschik, Koenraad Devriendt, Maria Teresa Dotti, Nursel Elcioglu, Blanca Gener, Timm O. Goecke, Małgorzata Krajewska-Walasek, Encarnación Guillén-Navarro, Joussef Hayek, Gunnar Houge, Esra Kilic, Pelin Özlem Simsek-Kiper, Vanesa López-González, Alma Kuechler, Stanislas Lyonnet, Francesca Mari, Annabella Marozza, Michèle Mathieu Dramard, Barbara Mikat, Gilles Morin, Fanny Morice-Picard, Ferda Özkinay, Anita Rauch, Alessandra Renieri, Sigrid Tinschert, G. Eda Utine, Catheline Vilain, Rossella Vivarelli, Christiane Zweier, Peter Nürnberg, Sven Rahmann, Joris Vermeesch, Hermann-Josef Lüdecke, Michael Zeschnigk, and Bernd Wollnik (Dec. 2013). "A comprehensive molecular study on Coffin–Siris and Nicolaides–Baraitser syndromes identifies a broad molecular and clinical spectrum converging on altered chromatin remodeling". *Human Molecular Genetics* 22.25, 5121–5135. DOI: 10.1093/hmg/ddt366.

Wilcoxon, Frank (1945). "Individual Comparisons by Ranking Methods". *Biometrics Bulletin* 1.6, 80–83. DOI: 10.2307/3001968.

Wong, Ka-Chun and Zhaolei Zhang (2014). "SNPdryad: predicting deleterious non-synonymous human SNPs using only orthologous protein sequences". *Bioinformatics (Oxford, England)* 30.8, 1112–1119. DOI: 10.1093/bioinformatics/btt769.

Wu, Hao, Tianlei Xu, Hao Feng, Li Chen, Ben Li, Bing Yao, Zhaohui Qin, Peng Jin, and Karen N. Conneely (July 2015). "Detection of differentially methylated regions from whole-genome bisulfite sequencing data without replicates". *Nucleic Acids Research* 43.21, gkv715. DOI: 10.1093/nar/gkv715.

Wu, Thomas D. and Serban Nacu (Apr. 2010). "Fast and SNP-tolerant detection of complex variants and splicing in short reads". *Bioinformatics* 26.7, 873–881. DOI: 10.1093/bioinformatics/btq057.

Xi, Yuanxin and Wei Li (2009). "BSMAP: whole genome bisulfite sequence MAPping program". *BMC Bioinformatics* 10, 232–239. DOI: 10.1186/1471-2105-10-232.

Xi, Yuanxin, Christoph Bock, Fabian Müller, Deqiang Sun, Alexander Meissner, and Wei Li (Feb. 2012). "RRBSMAP: a fast, accurate and user-friendly alignment tool for reduced representation bisulfite sequencing". *Bioinformatics* 28.3, 430–432. DOI: 10.1093/bioinformatics/btr668.

Zhang, Yan, Hongbo Liu, Jie Lv, Xue Xiao, Jiang Zhu, Xiaojuan Liu, Jianzhong Su, Xia Li, Qiong Wu, Fang Wang, and Ying Cui (May 2011). "QDMR: a quantitative method for identification of differentially methylated regions by entropy". *Nucleic Acids Research* 39.9, e58–e58. DOI: 10.1093/nar/gkr053.

Zheng, Xiaoqi, Qian Zhao, Hua-Jun Wu, Wei Li, Haiyun Wang, Clifford A Meyer, Qian Alvin Qin, Han Xu, Chongzhi Zang, Peng Jiang, Fuqiang Li, Yong Hou, Jianxing He, Jun Wang, Jun Wang, Peng Zhang, Yong Zhang, and Xiaole Shirley Liu (Aug. 2014). "MethylPurify: tumor purity deconvolution and differential methylation detection from single tumor DNA methylomes". *Genome Biology* 15.7, 419. DOI: 10.1186/s13059-014-0419-x.

ZURB, Inc. (2008). *Foundation*. https://foundation.zurb.com/.