SAARLAND UNIVERSITY
Department of Bioinformatics
MASTER'S THESIS

Alignment-free analysis of Whole Genome Bisulfite Sequencing data

submitted by
Olga Dikan
Saarbrücken
February 2023

Advisor:
Prof. Dr. Sven RAHMANN
Algorithmic Bioinformatics Group
Center for Bioinformatics, Saarland University
Saarbrücken, Germany

Reviewer 1: Prof. Dr. Sven RAHMANN
Reviewer 2: Prof. Dr. Fabian MÜLLER

Statement

I hereby confirm that this thesis is my own work and that I have documented all sources used.

Saarbrücken, den 06.02.2023

_____

List of Abbrevations

BS - bisulfite sequencing
BWT - Burrows-Wheeler transform
CpG - a cytosine nucleotide is followed by a guanine
DMR - differentially methylated regions
DNA - Deoxyribonucleic Acid
DNMT - DNA methyltransferase
FM-index - Full-text index in Minute space
GRC - Genome Reference Consortium
IGF2 - insulin-like growth factor 2
I/O - input/output
NGS - Next Generation Sequencing
PCR - polymerase chain reaction
SAM - Sequence Alignment/Map
TET - Ten-eleven translocation enzymes
WGBS - Whole Genome Bisulfite Sequencing
5mC - 5-Methylcytosine

Abstract

DNA methylation is one of the key mechanisms regulating the cell's properties and behavior. Disruption of normal methylation patterns leads to severe diseases, for example, cancer. Whole Genome Bisulfite Sequencing is a technique, which allows the identification of methylation patterns of a sample. There are several tools for performing methylation calling and other kinds of methylation analysis on WGBS data. However, all of these tools execute alignments of the WGBS sample's reads onto several regions of the reference genome, which is usually partially realized with the help of dynamic programming algorithms. To avoid this time- and memory-consuming step, we propose an alignment-free methylation calling method, based on the utilization of unique k-mers. The suggested approach served as the basis for a methylation calling tool, which produces results comparable to the ones obtained from modern tools, and, with the incorporation of further upgrades, could significantly outperform conventional applications.

Acknowledgements

To start with, I would like to pay tribute to my supervisor, Prof. Dr. Sven Rahmann, for all the supervision, troubleshooting and (let us be frank) patience that was given to me during the work on this project, which turned out to be more complicated, than what we initially had in mind.

Nothing of this would be possible without my wonderful family. The only reason I could get so far is their endless support and belief in me, especially my beloved brother (a.k.a. partner in crime and the closest friend), I am incredibly lucky to have all of you.

Ма, Па, всё это стало возможным только благодаря вам, вашей поддержке (во всех смыслах), вашей вере в меня, любви, заботе и усилиям. Я бесконечно вас люблю и скучаю по вам, и надеюсь, что скоро настанет мой черёд вас баловать и радовать!

And last but not least, I cannot express my gratitude enough towards the people that made the most comfortable social circle in my life: Joel, Polina, Lyuba, Swarna, Maciek, Chen, Alper, Anthony and Jeff (we miss you guys so much here!).

Little did I know, how extremely hard and unique in terms of experience times were awaiting for me, when I came to Saarbrücken to try to pursue the carrier of my dreams. And honestly speaking, without people mentioned here the time of my Master's studies would be just insufferable. Not everything went the best possible way for me, but thank to you, guys, I am looking into the future with as much hope, as I can afford myself in general. I wish to carry our relationships way further my graduation, and I hope that I have become your friend as you have become mine.

With all my heart

Contents

Chapter 1
Introduction

There is a significant amount of change that a human body is going to face throughout its lifetime. Some of those changes are normal, natural and healthy, others could lead to severe functional alterations and death. As one of the key processes shaping the behavior of human body cells we can certainly name DNA methylation, since it heavily influences the fact, which parts of a roughly 3 billion DNA base pairs long genome are going to be active in a particular cell at a certain moment, and which are not.

With that being said, doctors and researchers that are occupied with problems related to changes in cellular functions and properties are in need for reliable approaches for the identification of methylation patterns in patients' samples. From the chemical side the problem was solved with the development of the Whole Genome Bisulfite Sequencing technique (further addressed as WGBS). It produces a collection of short reads from the whole genome sample after sodium bisulfite treatment that provides clear markers for understanding the methylation state of DNA regions of interest.

However, WGBS specificity lies in reads being randomly allocated in genome, which requires the usage of special software tools for read mapping to the reference human genome for further analysis of the sample's methylation patterns. Over the last decades several powerful applications for WGBS data analysis were created, like Bismark (Krueger and Andrews, 2011), BS-Seeker2 (Guo et al., 2013), B-SOLANA (Kreck et al., 2011), Bison (Ryan and Ehninger, 2014), GNUMAP-bs (Hong et al., 2013), or Cloud-BS (Choi et al., 2018). While all of them have different procedure ideas in the base, there still exists one common step in those pipelines: alignments of the read's sequence to portions of the reference genome aimed to find

the most probable origin of the read, which is usually based on dynamic programming algorithms. Those make it the most time- and memory-consuming part of the process and the primary target for improvement for all new WGBS data analysis tools.

In this work, we would like to avoid the alignment procedure by introducing the usage of unique k-mers (k-letters long substrings seen only once in a sequence) for the identification of read's position in the genome and the methylation state of the regions of interest. Proposed further method could become the base for applications with potentially significantly reduced processing time, improved memory usage and results accuracy.

Chapter 2
Background

Before presenting the methodology we would like to provide some biological and computational aspects that served as the theoretical base for our tool.

2.1 Biological basics

2.1.1 Reference genome

The human reference genome represents a telomere-to-telomere assembly of a "universal" human genome, which is used further for annotating genes or comparing with the genome sequences of individual humans to search for pathogenic mutations and diverse elements (Wong et al., 2020). Being originated from the publicly funded Human Genome Project (Nurk et al., 2022), the first draft sequence covering most of the human genome was produced in 2000 (Consortium, 2001) and fully completed in 2003 (Wong et al., 2020). Nowadays the human reference genome assembly project is maintained by the Genome Reference Consortium (GRC) with its latest version (GRCh38) released in 2013 (Schneider et al., 2017). In this work, we used the most recent patch released in May 2022 (GRCh38.p14, Fig. 2.1) (available at `https://www.ncbi.nlm.nih.gov/grc/human`).

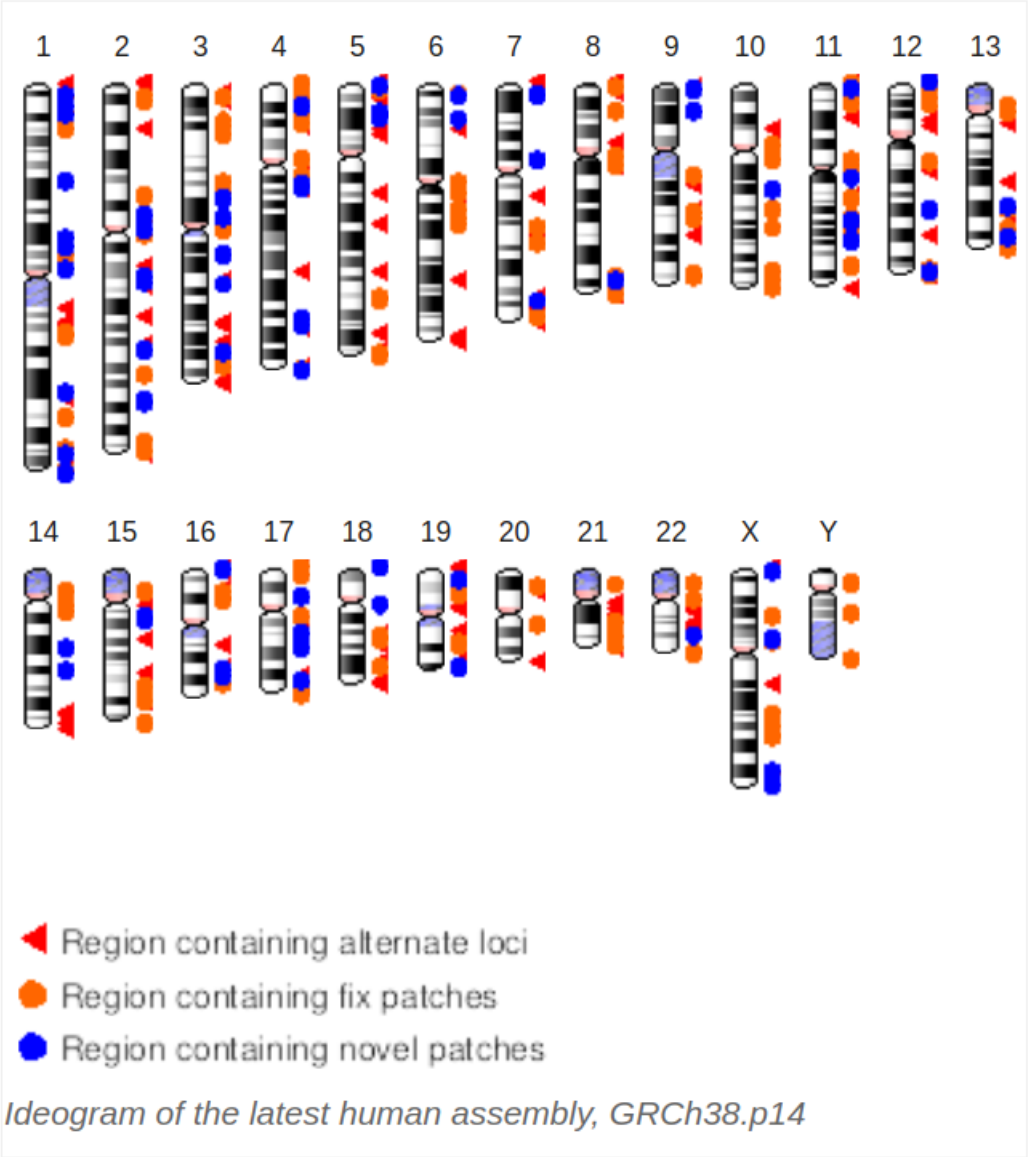

Figure 2.1: Ideogram of GRCh38.p14. Retrieved from GRC (Genome Reference Consortium, 2022). Fix patches represent changes to existing assembly sequences, novel patches represent the addition of new alternate loci to the assembly; according to `https://www.ncbi.nlm.nih.gov/grc/help/patches/`

2.1.2 DNA methylation

Mechanism

DNA methylation is a process of transferring a methyl group (CH_3) onto the 5-carbon position of the cytosine residue to form 5-methylcytosine (Fig. 2.2) (Moore et al., 2013).

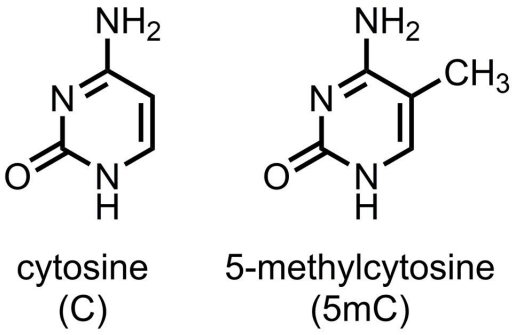

Figure 2.2: Cytosine and 5-methylcytosine. Retrieved from (Shi et al., 2017)

This is an epigenetic DNA performance influencing mechanism, which does not change the sequence itself, and its patterns are heritable. This reaction is performed by several kinds of DNA methyltransferases (DNMTs) that transport a methyl group from S-adenyl methionine (SAM) to the fifth carbon of a cytosine ring, resulting in the formation of 5mC (Moore et al., 2013). For the establishment of *de novo* methylation patterns are responsible DNMT3a and DNMT3b. DNMT1 is involved in the process of copying the methylation markers onto newly synthesized DNA strands during replication (Fig. 2.3).

CpGs and CpG islands

CpG dinucleotide is the part of the DNA sequence, where a cytosine residue is followed by a guanine one along the $5' \rightarrow 3'$ direction. CpG islands are the DNA regions with a high frequency of CpG occurrences. Although in mammalian organisms methylation occurs at cytosines in any genomic context, in somatic cells more than 98% of methylated cytosines appear in a CpG context (Jin et al., 2011). At the same time, CpG islands are typically associated with the promoter regions, which mark the start of a gene (Saxonov et al., 2006).

Role

The key role of DNA methylation lies in the epigenetic regulation of gene expression in the cell by engaging gene-repressing proteins or by hindering the binding of transcription factors at the promoter regions (Moore et al., 2013). In this light, the methylation state of CpG islands acquires exceptional importance as the hallmark of gene activity in the particular cell.

In the course of cell development, numerous amounts of methylation and demethylation processes drastically change the methylation patterns for cells to adapt their properties and functionality for their conditions. This leads to differentiated cells obtaining their stable and unique methylation patterns, which orchestrate tissue-specific gene transcription (Moore et al.,

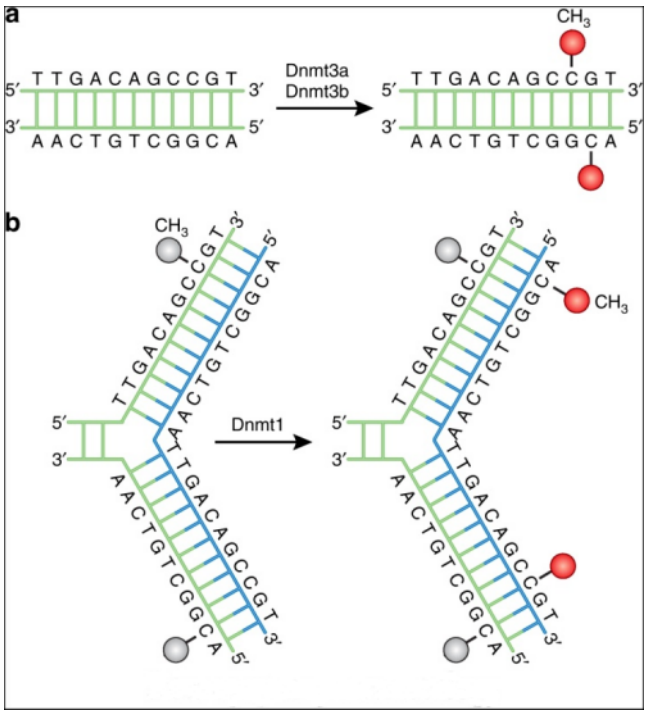

Figure 2.3: DNA methylation pathways. A family of DNA methyltransferases (Dnmts) catalyzes the transfer of a methyl group from S-adenyl methionine (SAM) to the fifth carbon of cytosine residue to form 5-methylcytosine (5mC). (a) Dnmt3a and Dnmt3b are the de novo Dnmts and transfer methyl groups (red) onto naked DNA. (b) Dnmt1 is the maintenance Dnmt and maintains the DNA methylation pattern during replication. When DNA undergoes semiconservative replication, the parental DNA strand retains the original DNA methylation pattern (gray). Dnmt1 associates at the replication foci and precisely replicates the original DNA methylation pattern by adding methyl groups (red) onto the newly formed daughter strand (blue). Retrieved from (Moore et al., 2013)

2013). Furthermore, hypermethylation silences the expression of certain genes, which inactivity is associated with aging, like estrogen receptors or IGF2 (Kandi and Vadakedath, 2015).

Maintainance of normal methylation patterns is crucial in cases of genomic regulation in the human body, such as genomic imprinting (parent-of-origin-specific allele expression), silencing of one copy of the X chromosome in females, suppression of repetitive element transcription (Jin et al., 2011). Therefore disruption of those patterns leads to severe genetic disorders.

One of the most important consequences of DNA methylation dysregulation is cancer. Various states can lead to cancerogenesis, to name a few: hypomethylation of imprinted loci or oncogenes, hypermethylation of tumor suppressor genes, DNMT mutations and expression misregulations, dysregulation of TETs (Jin and Liu, 2018). Moreover, primary tumor and

distant metastases in the same individual have shown different states of epigenomic reprogramming (McDonald et al., 2017), which could be further exploited by specialists for diagnosis and treatment. Apart from cancer, DNA methylation has been shown to play a leading role in a range of common diseases, such as autoimmune diseases, metabolic and neurological disorders, Huntington's and Alzheimer's diseases, and many more (Jin and Liu, 2018).

2.1.3 WGBS

Bisulfite sequencing is a technique developed by Frommer et al. (Frommer et al., 1992), which is based on the made in 1970 discovery that treating DNA with sodium bisulfite deaminated cytosine residues into uracil ones (Harrison and Parle-McDermott, 2011). The suggested protocol utilized the finding that 5mCs under sodium bisulfite treatment were converted to uracils at much slower rates, than unmethylated cytosines. As a result, during the subsequent DNA sequencing procedure the uracils originating from unmethylated cytosines were recognized as thymines, whereas 5-methylcytosines stayed cytosines. This transformation allows to recognize the methylation state of concrete cytosines by comparison of the obtained after treatment reads to the reference genome (Fig. 2.4).

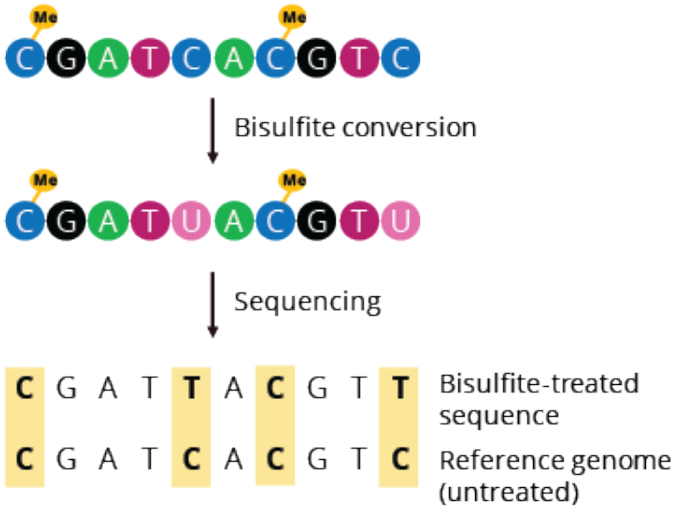

Figure 2.4: Schematic diagram of bisulfite sequencing workflow. Retrieved from `https://nbis-workshop-epigenomics.readthedocs.io/en/latest/content/tutorials/methylationSeq/Seq_Tutorial.html`

Whole Genome Bisulfite Sequencing arose later as a combination of bisulfite treatment step and next-generation sequencing techniques, for example, shotgun sequencing (Adams, 2008), which consists of sequencing many small, overlapping, and randomly allocated DNA fragments (reads) with further assembling of reads into larger contigs and, finally, chromosomes (Fig. 2.5).

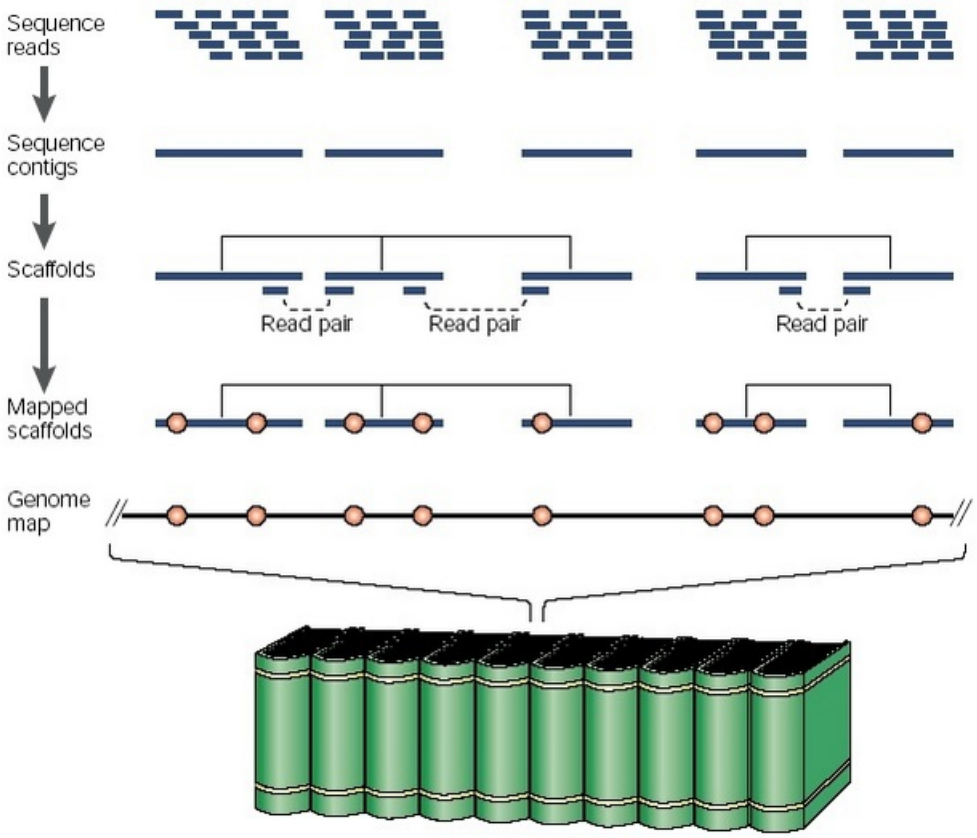

Figure 2.5: Long-range sequence assembly in whole-genome shotgun sequencing. Individual sequence reads generated in a whole-genome shotgun-sequencing project are initially assembled into sequence contigs. Groups of sequence contigs are then organized into scaffolds based on linking information provided by read pairs (in each case, with one sequence read from a pair assembling into one contig and the other read into another contig). In turn, the scaffolds can be aligned relative to the source genome (represented by an encyclopedia set) by the identification of already mapped, sequence-based landmarks (for example, STSs, genetic markers, and genes; depicted as red circles) in the sequence contigs, thereby associating them with a known location on the genome map. Retrieved from (Green, 2001)

At present, WGBS is considered to be the gold-standard technology for the detection of DNA methylation state, because it provides a qualitative, quantitative and efficient approach to discriminate between cytosine and 5-methylcytosine at single base-pair resolution (Li and Tollefsbol, 2011).

2.2 Computational basics

2.2.1 K-mers

Generally speaking, k-mers are sequences of length k, composed from letters of a previously defined alphabet. By the term *k-mers* could be meant both all possible sequences of length k on a fixed alphabet and all the substrings of length k from a given sequence. In a bioinformatical context, the alphabet can represent DNA nucleotides (A, C, G, T), RNA nucleotides (A, C, G, U), or amino acids (their 1-letter symbols). A particular k-mer is called *unique* in the context of a sequence if it appears only once in the sequence of interest (Fig. 2.6).

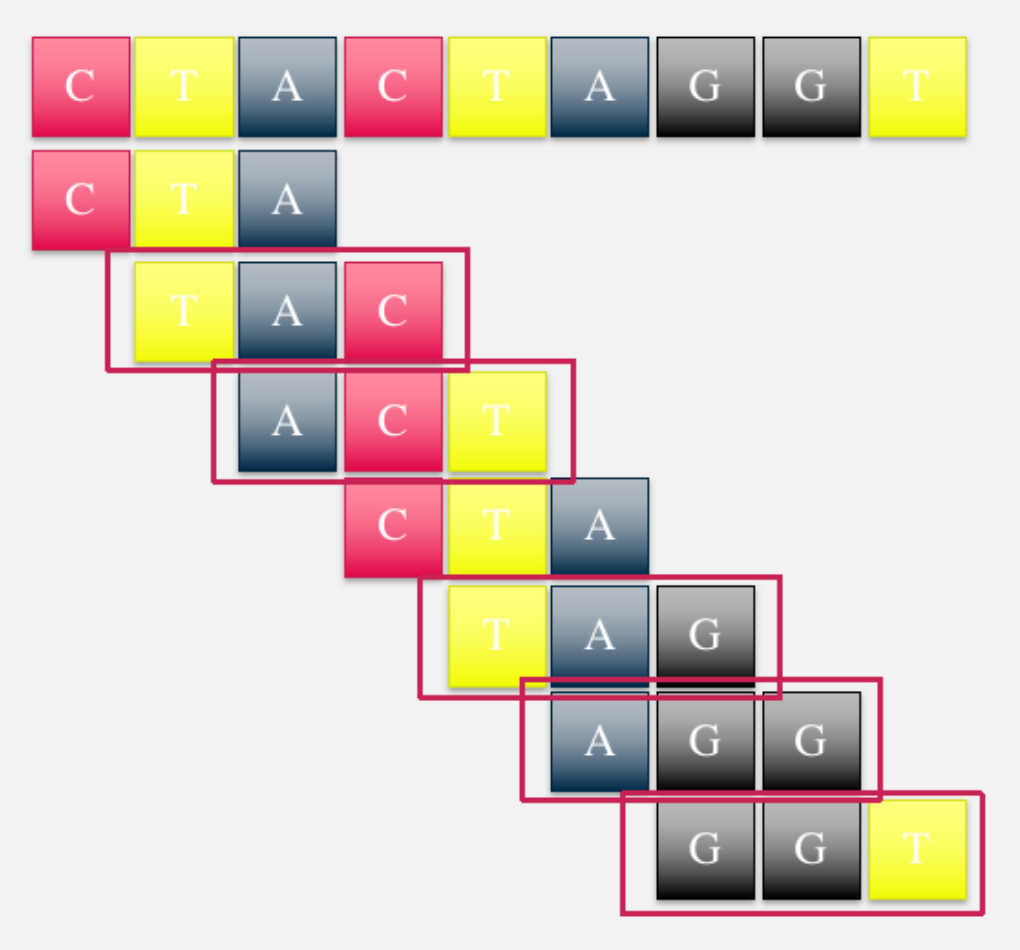

Figure 2.6: K-mers of an exemplary sequence. Red frames identify the *unique* k-mers of the sequence.

Regarding applications of k-mers in bioinformatics, one should note sequence assembly based on De Bruijn graphs (Li et al., 2010), *de novo*

detection of repeated sequences such as transposable elements (Price et al., 2005), DNA barcoding of species (Meher et al., 2016), identification of mutation or polymorphism using next-generation sequencing data (Nordström et al., 2013) and identification of biomarkers for diseases from samples (Wang et al., 2018).

2.2.2 Numerical encoding of k-mers

Quite often bioinformatical problems related to sequence analysis demand many k-mers from DNA or RNA sequences to be produced and stored for further analysis steps. In such setting, some k-mer compression is desirable to not exhaust memory resources. One of the methods for compression of k-mers from DNA/RNA sequences is numerical encoding, which translates each letter of the alphabet into a 2-bit number (namely A is encoded as 0, C as 1, G as 2, and T/U as 3). In a k-mer all the letters are replaced by their 2-bit codes, the result is the bit representation of a number, which becomes the code of this particular k-mer. Such transformation not only assigns a unique code to each k-mer, but also makes it possible to precisely retrieve the actual k-mer from its code if needed.

2.2.3 K-mer counting tools

Originally the problem of counting occurrences of different k-mers in a sequence is quite straightforward, for the cases of small sequences and ks it is enough to implement the counter as a dictionary with k-mers being the keys, and their occurrences being the values. However, in a setting of scientific research, it is usually required to analyze large sequencing outputs (eg. whole genome sequencing with an average coverage of 70) with k-mers of length between 10 and 30. For the tasks with the required value of k >15, the naive implementation becomes excessively memory- and computationally expensive. With the rise of next-generation sequencing and the greatly increased amount of sequencing data retrieved for further analysis, the problem of efficient k-mer counting became rather acute.

To address this matter, a number of k-mer counting tools were developed, adopting various approaches for reducing processing time and memory usage. One of the first attempts was Jellyfish (Marçais and Kingsford, 2011), which introduced the usage of a lock-free hash table as a counter structure with the ability to insert k-mers and update counts in parallel by multithreading. Moreover, it allowed transferring of the hash table from memory onto disk at the moment it got full.

The next in line were tools based on the usage of disk space to temporarily store k-mers, while they are processed in smaller portions in memory. With a payoff in increased I/O costs, the disk-based tools significantly reduced memory usage, which enabled the processing of large genomic data with non-specific hardware. One of the most popular software of this type is KMC3 (Kokot et al., 2017), which applies the ideas of minimum substring partitioning for reducing I/O operations and sorting of k-mers for fast counting (all the occurrences of a k-mer are located adjacently, which makes them easy to count).

The latest generation of k-mer counting tools is marked by the development of Gerbil (Erbert et al., 2017), which combines efficient disk utilization for k-mers storage, exploitation of hash tables for counting purposes, and multiple threading for hardware usage optimization. Furthermore, it allows the usage of GPU for additional acceleration of computations.

More detailed comparison and analysis of k-mer counting tools can be found in the following work (Manekar and Sathe, 2018).

2.3 Existing methylation analysis tools

This section is purposed to provide insights into the current state of methylation analysis tools.

2.3.1 Tools for methylation calling (using mapped reads)

The pioneering software for DNA methylation calling was Bismark (Krueger and Andrews, 2011), which adapted Bowtie (Langmead et al., 2009) technology for its alignment stage.

Bowtie

Bowtie is an alignment tool that specializes in the mapping of short reads from DNA sequences to large genome ones. To achieve optimal performance in terms of speed and memory requirements, it exploits Burrows-Wheeler transformation (Burrows and Wheeler, 1994) in pair with the full-text minute-space index (Ferragina and Manzini, 2000). BWT is a reversible transformation of a string, obtained by sorting all the cyclic rotations of this string (with appended symbol \$) and assembling their last symbols in a resulting transformation string. The FM index searching algorithm proposed by Ferragina and Manzini allows to search occurrences of a substring based on the BWT of the string, but is suited just for finding exact matches. To make it suitable for operating with sequencing data, which needs room for sequencing errors and genetic variations, Bowtie developers introduced several updates for allowing reasonable mismatches,

while maintaining the quality of short read alignments.

Bismark

According to WGBS protocol, after exposing the sample DNA to the sodium bisulfite treatment, a PCR amplification procedure is performed and a collection of short reads is obtained. However, there is no certain way to tell beforehand, whether each particular read takes its origin from the forward DNA strand or its reverse complement. To overcome this obstacle and gain accurate results, Bismark first aims to find the exact origin of the read, considering all possible scenarios.

The process outline looks as follows:

1. Obtaining the C-to-T (all cytosines are replaced with thymines) and G-to-A (all guanines are replaced with adenines, corresponds to the C-to-T transformation on the reverse strand) transformations of the bisulfite read.
2. 4 parallel Bowtie alignments (the C-to-T version of bisulfite read is aligned to the C-to-T transformations of forward and reverse strands of the reference genome, likewise for the G-to-A version of bisulfite read) to find the unique best match.
3. The original bisulfite read is compared to the strand of origin at the aligned region to retrieve the methylation calls for the cytosines (or, depending on the context, guanines) of the read (Fig. 2.7).

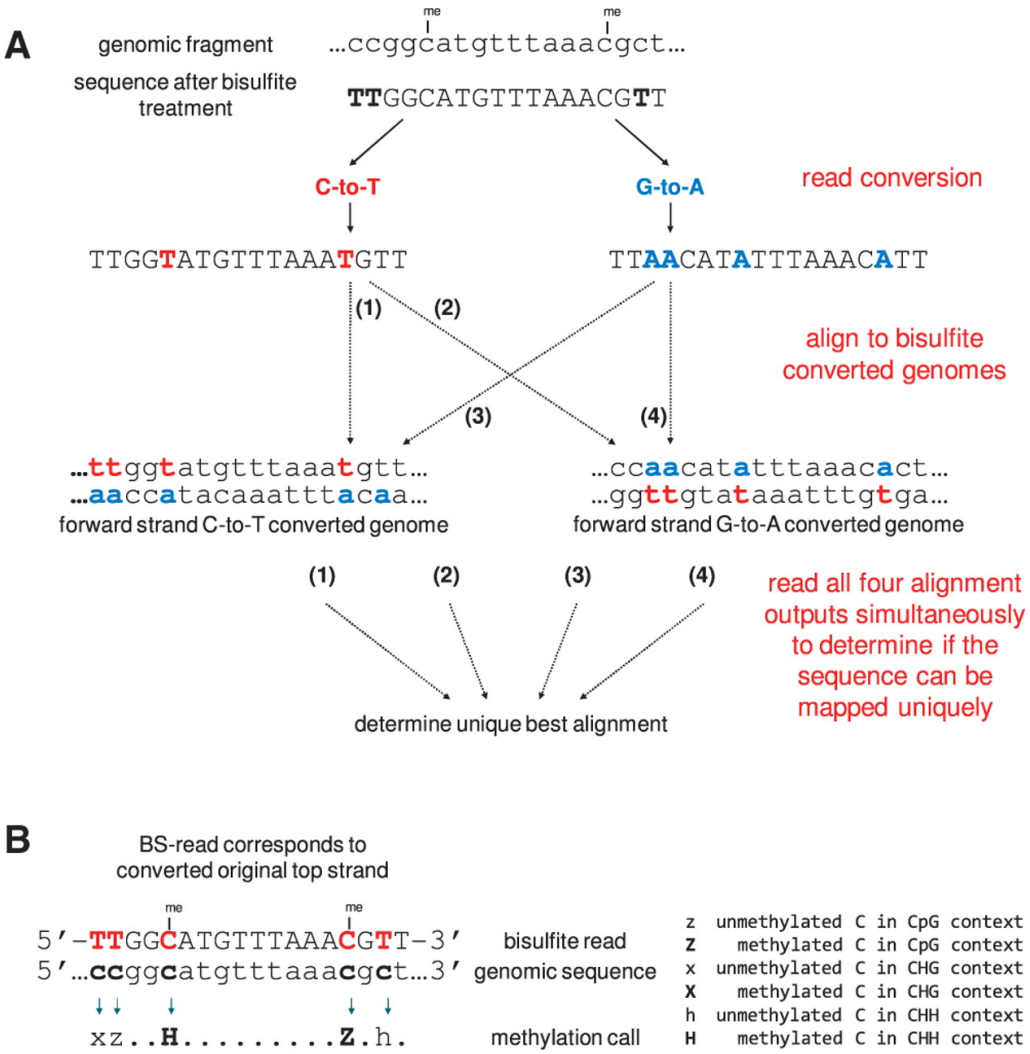

Figure 2.7: Bismark's approach to bisulfite mapping and methylation calling. Retrieved from (Krueger and Andrews, 2011)

Software for analysis of WGBS data is an actively evolving field with a variety of new interesting approaches arising, especially for the alignment step. Beginning with more classical BS-Seeker2 (Guo et al., 2013) originating from analogous to Bismark BS-Seeker, but improved with Bowtie2 (Langmead and Salzberg, 2012) - the updated version of Bowtie, allowing local and longer-gapped alignment, one can find BS data analysis tool for any taste, task and amount of available resources.

For example, Bison (Ryan and Ehninger, 2014) makes use of multiple nodes of a computer cluster for alignment optimization, whereas GNUMAP-bs (Hong et al., 2013) tries out a probabilistic algorithm approach and Cloud-BS (Choi et al., 2018) exploits the modern cloud infrastructure for parallel alignment execution on distributed nodes.

2.3.2 Tools for bisulfite-aware read mapping

To speed up the alignment process and make it more accurate on WGBS data, classical aligners have sought to be transformed into dedicated bisulfite-aware read mapping tools, for example, bwa-meth (Pedersen et al., 2014).

bwa-meth

Another short read alignment tool based on BWT was presented in 2009 bwa (Li and Durbin, 2009), which supported reads from different sequencing machines, such as Illumina and AB SOLiD, and produced output in the new standard SAM format. When the standard length of sequenced reads increased threefold, the end-to-end (each base of the read) alignment lost its relevance to the local (partial) alignment and accounting for the longer gaps. Under these circumstances, bwa evolved to bwa-mem (Li, 2013), which later served as the foundation for bwa-meth (Pedersen et al., 2014) - specialized 3-base aligner for modern bisulfite reads, which streams bisulfite reads conversions directly to the aligner instead of writing temporary files on the disk.

2.3.3 Integrated pipelines

Since Bismark was developed and published in 2011, the world of NGS technology has changed rapidly. The growing length of sequenced reads, various types of sequencing machines, and newly developed protocols for analysis of methylation data required more complex tools, producing all possible kinds of metrics from the higher amount of data with the most optimal speed and memory utilization in one run. The start of the new era was noted with the creation of pipelines - software allowing the usage of a collection of tools for consecutive processing of the input data, seamlessly passing outputs of one tool as the inputs of the following ones.

One of the most recent software of this kind is wg-blimp (Wöste et al., 2020) - targeted for analysis of the WGBS data pipeline, which includes specialized aligner bwa-meth.

wg-blimp

In comparison with Bismark being only the tool for bisulfite reads alignment and methylation calling, the modern pipeline wg-blimp performs multilateral methylation analysis of provided as input FASTQ files with WGBS reads and the reference genome. Constructed with the help of workflow engine Snakemake (Köster and Rahmann, 2012), wg-blimp combines tools for alignment of bisulfite reads to the reference genome, deduplication of the reads, methylation calling, quality control, DMR calling and methylome segmentation. All the produced reports can be inspected with

a special web interface (Fig. 2.8).

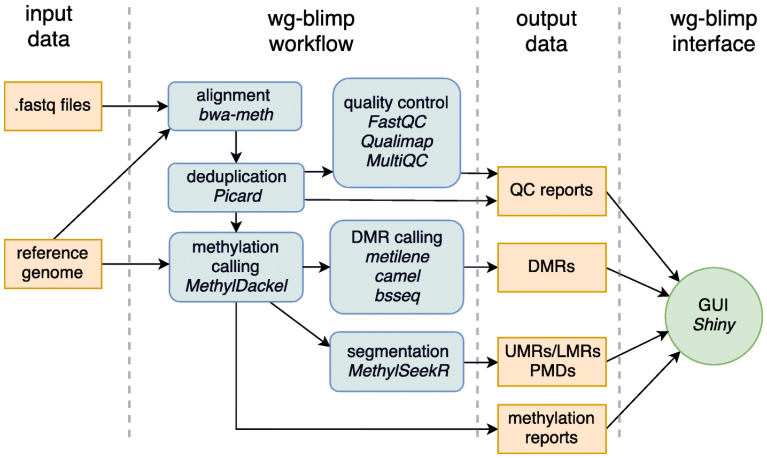

Figure 2.8: wg-blimp workflow overview. Retrieved from (Wöste et al., 2020)

Chapter 3
Methodology and implementation

As mentioned earlier, all the tools for methylation calling on WGBS data include in their workflows alignment step, which is the most significant and resource-requiring part of the process. We try to address the issue of resource optimization by replacing the alignment procedure with a new method for CpG identification in a read, based on the unique k-mers utilization.

3.1 Methodology

3.1.1 Idea

Human genome has approximate size of 2^{32} basepairs, which is approximately 4^{16} basepairs. Considering that each basepair from the DNA nucleotide alphabet (A, C, G, T) has the same probability to occur at a certain DNA position, then by taking $\log_4 n$ (with n being the sequence length) we can find the size k for k-mers to appear approximately once in the sequence. For the human genome, the k would be 16. This means that when taking k > 16 we will observe the bigger portion of k-mers being non-present or occurring only once in the genome sequence with the rise of k.

The base idea for the method lies in finding unique k-mers in the context of CtoT, GtoA transformations of the reference genome and their reverse complements in ± 50-window around each CpG position and storing them as "marker" k-mers for those CpGs in chromosome-wise lists together with the shift value (difference in starting position of the k-mer and CpG

position in the forward direction). Later the WGBS data reads are going to be split into the k-mers, those k-mers will undergo CtoT and GtoA transformations and be compared to the "marker" k-mers of the chromosome of interest. If matches are found, the actual CpG position in the read is calculated with the help of shift value, and in case it is a position inside the read, the methylation call is performed for that read position.

Choosing k high enough will provide us with a higher number of unique k-mers (potentially) around the associated CpG positions, which gives us more possibilities to still being able to identify a CpG position in case of mutations disrupting some of the "marker" k-mers. Additionally, the reasonable upper limit for the size of k would be 32, for the actual k-mer being able to transform into a number (taking 2 bits to code each symbol) and store it in a numpy uint64 variable. We chose the k to be 29 since it seems to provide an optimal balance between the number of unique k-mers found in the transformed variations of the genome, and the amount of data to store for the method realization.

3.1.2 Reasoning

Before starting to work on the method's implementation, we decided to validate the potential efficiency of such an approach. For that we generated the following statistics: we checked the percentages of the CpG positions having 0 and 5 or more unique 29-mers (in the context of reference genome sequence being transformed) in ± 50-windows around them chromosome-wise. The first case accounts for basically invisible to our tool CpG positions, the second one shows how many of CpGs are almost certainly identifiable with the suggested approach (Fig. 3.1).

Keeping in mind that the plot was generated just for the unique 29-mers corresponding to the forward CtoT transformation, we can already see a promising picture for each chromosome (except for Y), with the percentages of lost CpG positions around 15% on average, and not less, than 70%, and in many cases even over 80% of CpG positions almost surely visible by the tool.

3.1.3 Reverse complements

To account for the possibilities of the WGBS reads being produced from the reverse DNA strands, we included the reverse complements to the CtoT and GtoA transformed forward strands in the process of obtaining the unique 29-mers and forming lists of "marker" 29-mers.

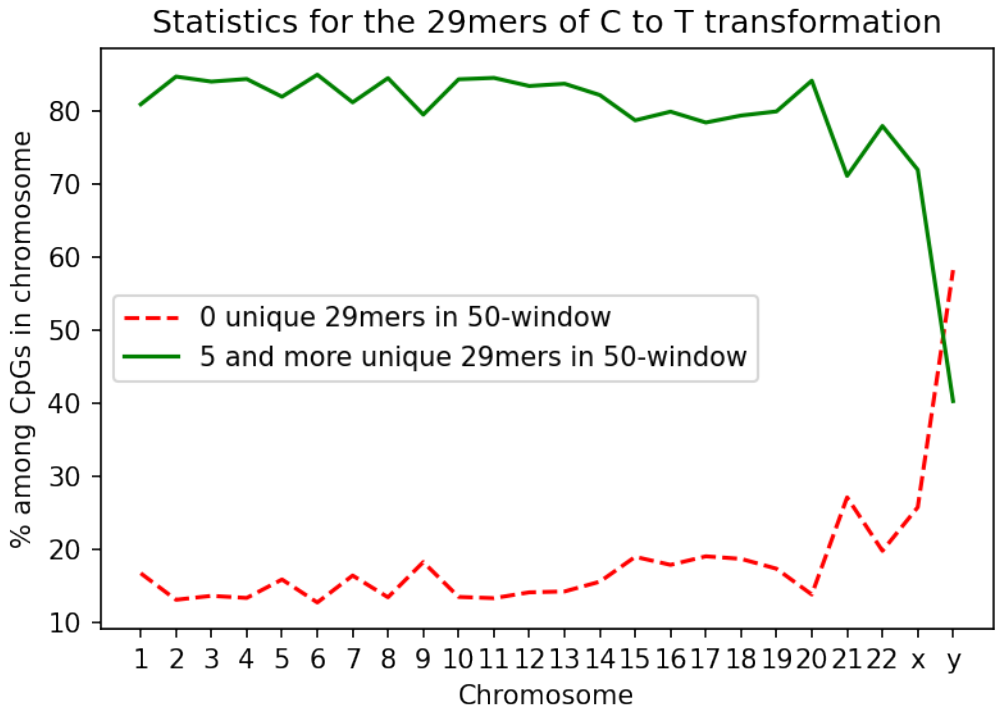

Figure 3.1: Statistics for the unique 29-mers abundance in ± 50-window around CpG positions, chromosome-wise. 29-mers are considered unique in the context of CtoT and GtoA transformations of the forward DNA strand and their reverse complements; statistics were retrieved for the 29-mers corresponding to the CtoT transformation of the forward DNA strand.

3.2 Workflow

In this section, I am going to show in the sequence of diagrams, how the key processes in this work are structured.

3.2.1 Generation of input files

The process of the main script's input files generation from the given reference genome sequence is depicted in Fig. 3.2.

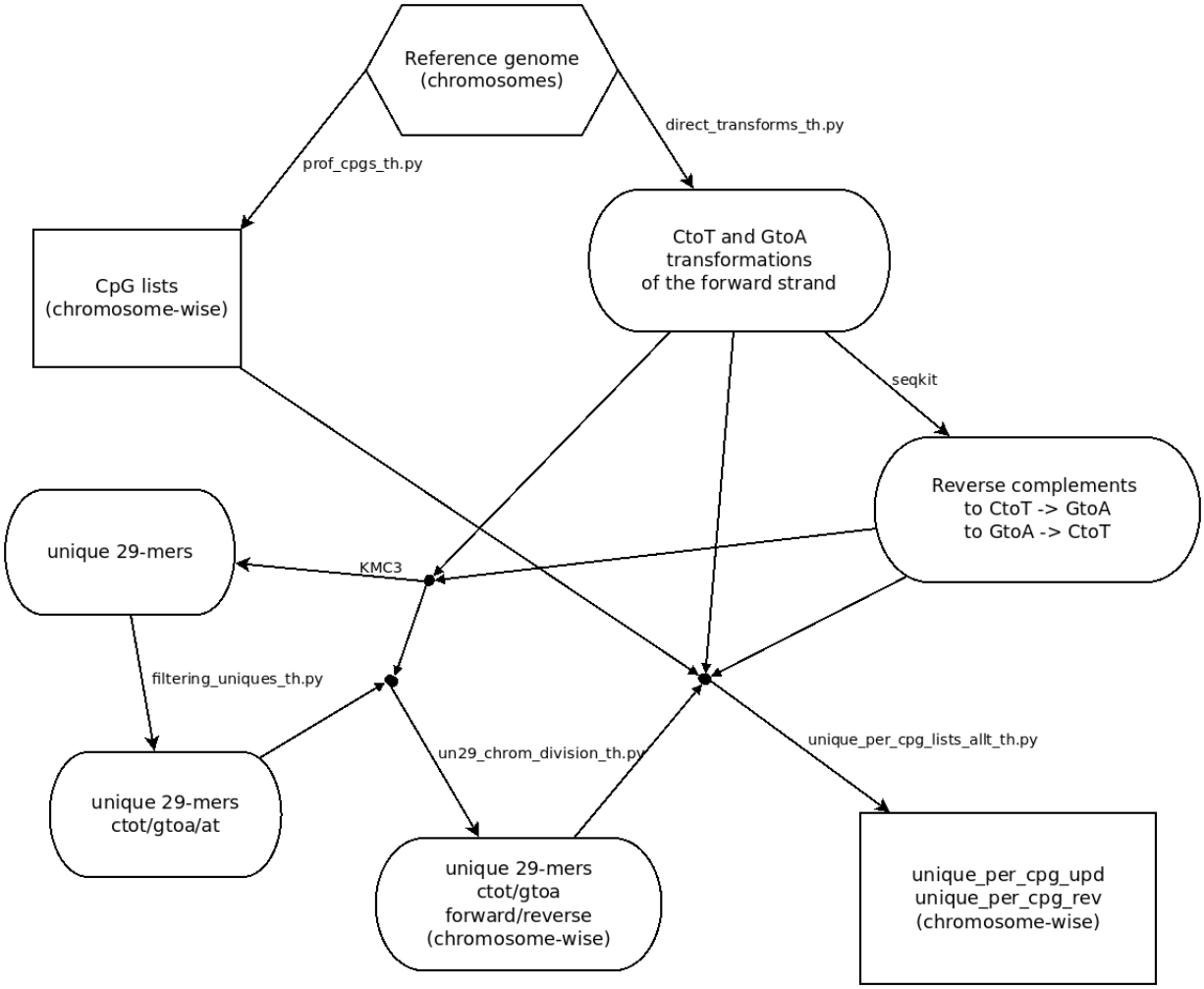

Figure 3.2: Generation of input files. Near the arrows are noted the scripts and software used at the particular step. Dots represent joint input of incoming into that dot files for the script/software, noted at the outcoming arrow. Hexagon depicts the input files, rectangles note the desired outcome files, and rounded rectangles show intermediate files.

As the input files, we take the human reference genome split into chromosomes, the resulting files are the lists of CpG positions for each chromosome and the lists of "marker" unique 29-mers, with associated with them CpG positions and values of shift between the CpG position and the starting position of the 29-mer (calculated with respect to the strand). After obtaining CtoT and GtoA transformations of the forward strand DNA sequences, we also generate the reverse complements to the first, which correspond to the GtoA and CtoT transformations of the reverse DNA strand respectively. Then we get unique in the context of all 4 sequences 29-mers, which we split into the ones that correspond to the CtoT and GtoA transformations, and the ones that only have adenines and thymines in them, so we cannot tell for sure, during which transformation they were generated. Afterwards, we form for each chromosome 4 files with unique 29-mers that appear in CtoT and GtoA transformations of both forward and reverse strands of the chromosome. Later with the help of those, lists of CpG positions and the actual sequences of the transformations we get the lists of the "marker" 29-mers.

3.2.2 Input structures

Fig. 3.3 shows, which structures are built from the previously generated input files at the beginning of the main script's execution.

3.2.3 General workflow of the method

In Fig. 3.4 we can trace the overall workflow of the main script's execution.

For each file in the sample directory, we retrieve all the reads. Then the reads are split into approximately equal parts, and each of those is processed by its separate process. After processing the portion of reads, intermediate statistics dictionaries update the cumulative one for the whole directory, which is later written into the output file.

3.2.4 Read processing

A more detailed description of one read's processing procedure is presented in Fig. 3.5.

From a read we obtain all the 29-mers, then for each 29-mer we retrieve the CtoT and GtoA transformations, which are converted to numerical codes. Then each version is checked to appear among the indices of the searching structure. If the match is found, we get all the CpG positions and shifts, associated with the 29-mer. Afterwards, each CpG is checked, whether it has not already been seen in this read, and whether it is inside

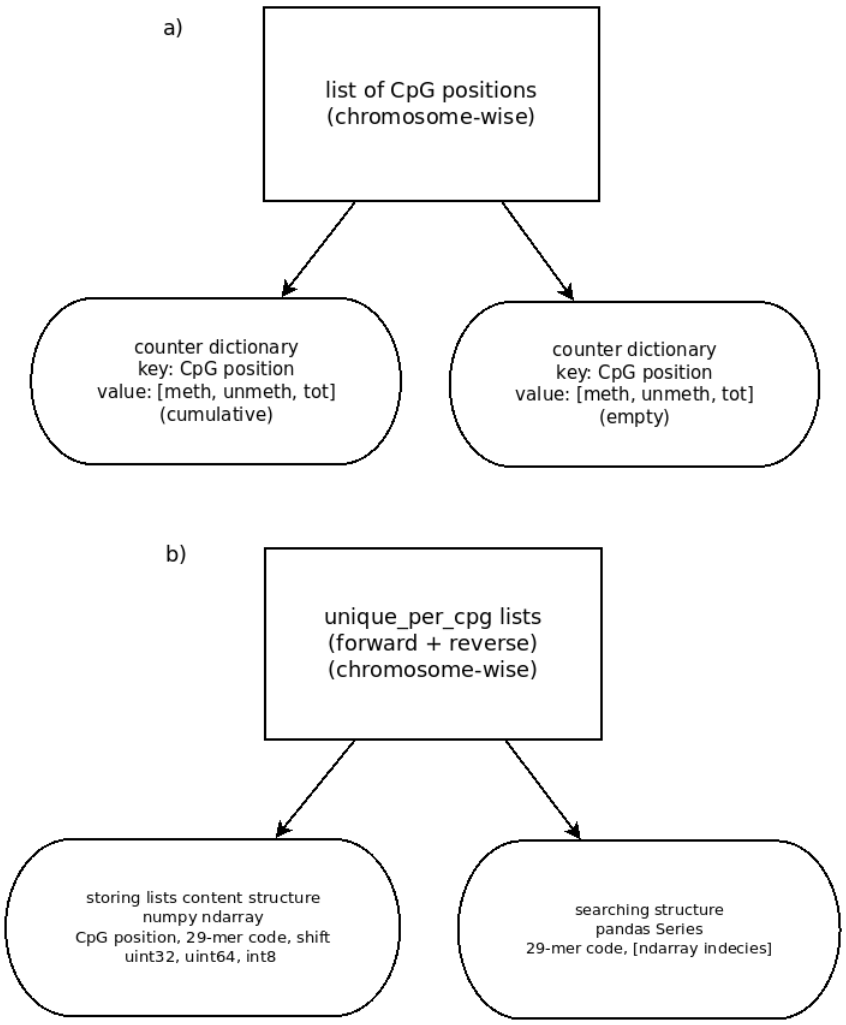

Figure 3.3: Structures generated from input files. a) From the list of CpG positions we generate 2 counter dictionaries, both having CpG positions as keys, and storing an array with 3 counters (for methylated, unmethylated, and total appearances of the CpG) as values. One of the dictionaries is used for accumulating statistics across the whole sample directory, the other stays empty to be passed for the collection of intermediate statistics for the current portion of reads. b) From the unique_per_cpg lists we generate 2 structures, one is used for storing all the information from files in the machine's memory, and the other is dedicated to the efficient search in the previous one. The storing structure is the numpy ndarray, where the values are the arrays with CpG position (uint32), the associated "marker" 29-mer converted into a numerical code (uint64), and the shift value (int8). The searching structure is a pandas Series with all the 29-mers codes being the indices, and the arrays of the indices of all appearances of the corresponding 29-mer codes in the storing ndarray being the values.

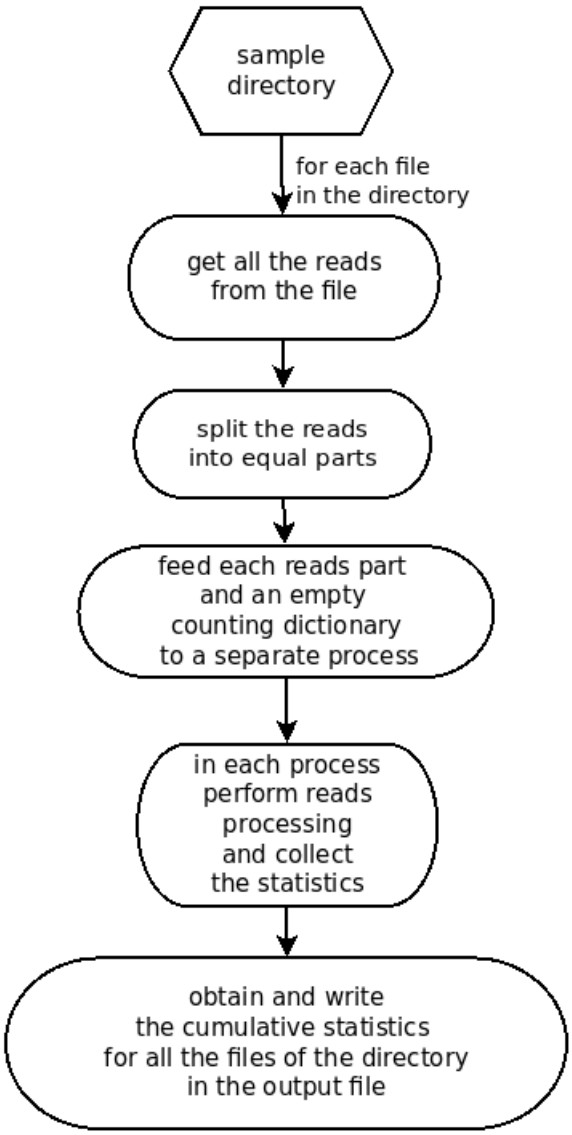

Figure 3.4: General workflow.

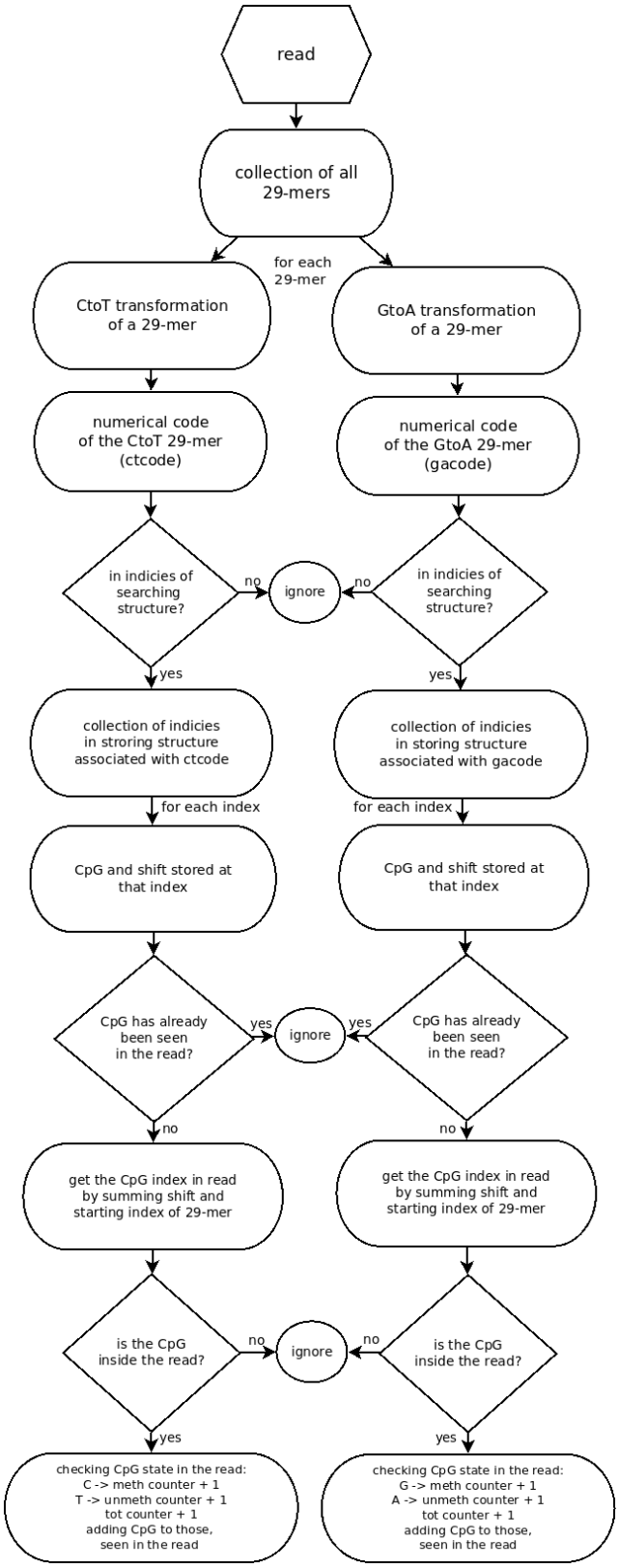

Figure 3.5: Processing of one read.

the read. If both conditions are met, the methylation call is performed and the result is stored in the intermediate statistics dictionary.

3.3 Implementation

In this part, I would like to give some comments about the choices for software, structures, and principles used in the course of the work.

seqkit

For getting the reverse complement sequences for the CtoT and GtoA conversions of the forward DNA strand the tool seqkit (Shen et al., 2016) was used. It provides an extensive command pool for efficient parsing of FASTA/FASTQ sequences and is presented in a cross-platform version with a simple installation procedure.

KMC3

KMC3 (Kokot et al., 2017) was described in the ***Background*** section. This k-mer counting software provides multi-platform executables for smooth installation, is time- and memory usage optimized, and allows to solve highly customized k-mer counting tasks with a wide range of settings options. Moreover, it makes retrieving all the actual k-mers sequences of interest possible.

numerical codes

Usage of the 29-mers in sequence format (both string and binary versions) requires large memory space. As a method of compression, we decided to convert the 29-mers strings into their unique numerical codes, which discriminate between the one-directional 29-mer and its reverse complement (assign different codes to them). For this conversion, the code from **hackgap** project of the Rahmann Lab (`https://gitlab.com/rahmannlab`) was used.

storing structure

As a storing structure for the contents of the lists of "marker" 29-mers was chosen numpy ndarray with preset values of the second dimension arrays fields. The *loadtxt* method provides a faster reading of data from file, than conventional python file reading, moreover, it allows to set the types of fields already during the reading from the file. Furthermore, numpy ndarray provides more compact data storage, than the usual 2-dimensional array and has a convenient method *array_split* for partitioning the array into equal chunks. To make it even more space-efficient, the types of fields were carefully chosen based on the range of possible values of each field.

searching structure

Since CpGs can be positioned pretty close to each other in some cases, one "marker" 29-mer can be associated with more, than one CpG. When a "marker" 29-mer is spotted among the 29-mers of the read, we want to get information on the methylation state of as many CpGs, as this particular "marker" 29-mer can identify in the read. This means that we need to retrieve all the CpG positions and shift values associated with the "marker" 29-mer from the storing ndarray each time we find a match among the 29-mers of the read. To make this search time-efficient, we construct a searching structure, which is of pandas Series type. All the numerical codes of the "marker" 29-mers in the storing ndarray serve as the structure's indices, and as the values are written arrays, containing the ndarray indices of all appearances of the corresponding index code of the "marker" 29-mer. A combination of the hashed search among the pandas Series indices and the index element access of the ndarray provides us with a fast and robust read processing performance. As will be shown later, the most execution time is spent at the searching structure construction step, which is balanced out by stable and calculatable sample files processing time that depends only on the number of files in the sample directory of interest and the number of processes used at the multiprocessing step of the procedure.

counter structure

For the methylation statistics collecting structures are used python dictionaries with keys being the CpG positions of the chromosome of interest and the values being arrays of counters, initially set to zeros (for methylated, unmethylated, and total occurrences of the CpG). Dictionaries secure the efficient search among the keys and simplicity in collecting the statistics for the reads batches and updating the cumulative appearances values for the whole sample directory.

multiprocessing

To accelerate the reads processing part, ProcessPoolExecutor from concurrent.futures module is exploited. With the rise of the number of processes used, the execution time for one file of the sample directory decreases, being initially around 1.5 hours with 1 process, but already dropping to 45 minutes with 2 processes.

snakemake

Workflow execution management is realized with help of snakemake (Köster and Rahmann, 2012). With the main working script performing methylation calling of one chromosome of interest for one sample directory, snakemake makes flexible execution options available. The current version has two working modes: one for processing one sample directory

for multiple chromosomes, and the other for processing multiple sample directories for one chromosome. All the necessary directories paths can be changed in the Snakefile, and the concrete execution of modes can be tailored through bash scripts to meet the user's needs. After that, the execution of the program requires just the snakemake command.

Chapter 4
Computational experiments

This chapter is dedicated to the description of the method's current performance and the comparison of our results to those obtained by wg-blimp.

4.1 Sequence data quality

Here we provide a table with chromosome-wise data on chromosome length and amount of non-ACGT bases (Table 4.1).

chromosome	length (bp)	non-ACGT (% from length)
Y	57227415	30812372 (53.84)
X	156040895	1147866 (0.74)
22	50818468	11658691 (22.94)
21	46709983	6621364 (14.17)
20	64444167	499910 (0.78)
19	58617616	176858 (0.30)
18	80373285	283680 (0.35)
17	83257441	337237 (0.40)
16	90338345	8532402 (9.44)
15	101991189	17349864 (17.01)
14	107043718	16475569 (15.39)
13	114364328	16381203 (14.32)
12	133275309	137493 (0.10)
11	135086622	552880 (0.41)
10	133797422	534460 (0.40)
9	138394717	16604167 (12.00)
8	145138636	370500 (0.25)
7	159345973	375842 (0.24)
6	170805979	727457 (0.43)
5	181538259	272881 (0.15)
4	190214555	461888 (0.24)
3	198295559	195424 (0.10)
2	242193529	1645301 (0.68)
1	248956422	18475410 (7.42)

Table 4.1: Sequence data quality table. For each chromosome given its length and number of non-ACGT characters in the sequence, which represent sequencing uncertainty.

This table shows that 62.5% chromosomes already have a very good sequence quality (with uncertainty percentage < 1%), and for those chromosomes we already have a good understanding regarding the abundance of unique k-mers in the proximity of CpG positions. For the rest chromosomes, we can expect an increase in both the amount of CpG positions and the number of "marker" unique k-mers with the growth of their sequencing quality.

4.2 K-mer statistics

This section provides some insights regarding the statistics connected to k-mers.

Chromosome-wise k-mer numbers

In this part, we present tables with the number of k-mers in total and unique in particular for k values of 25 (Table 4.2), 27 (Table 4.3), and 29 (Table 4.4). The k-mers were counted in the context of 4 transformation sequences for each chromosome: forward strand CtoT and GtoA transfor-

mations, as well as their reverse vis-a-vis.

chromosome	total 25-mers	unique 25-mers (% from total)
Y	105654556	69033662 (65.34)
X	619569092	463031316 (74.73)
22	156634500	117453206 (74.98)
21	160349580	126069430 (78.62)
20	255768484	206089110 (80.58)
19	233762264	161042584 (68.89)
18	320353916	258963234 (80.84)
17	331676336	244437034 (73.68)
16	327221756	248151674 (75.84)
15	338563652	262965858 (77.67)
14	362270292	294278350 (81.23)
13	391930480	328905052 (83.92)
12	532548480	424344616 (79.68)
11	538133432	429158738 (79.75)
10	533044368	428402400 (80.37)
9	487157880	373083462 (76.58)
8	579071488	471766246 (81.47)
7	635878700	494453234 (77.76)
6	680312648	552049498 (81.15)
5	725058056	578056614 (79.73)
4	759009036	613907560 (80.88)
3	792397852	639093304 (80.65)
2	962189828	770850436 (80.11)
1	921910308	709588616 (76.97)

Table 4.2: 25-mers table. For each chromosome given the numbers of total and unique 25-mers (with the percentage from the total number). The 25-mers were counted in the context of 4 transformations of chromosome sequence.

chromosome	total 27-mers	unique 27-mers (% from total)
Y	105654092	71165848 (67.36)
X	619568844	482027452 (77.80)
22	156634116	120494228 (76.93)
21	160349172	128800868 (80.32)
20	255767772	211454528 (82.67)
19	233762200	167113018 (71.49)
18	320353572	265458372 (82.86)
17	331675968	251985244 (75.97)
16	327221588	255464824 (78.07)
15	338563516	270390594 (79.86)
14	362270100	302876602 (83.60)
13	391930312	338152852 (86.28)
12	532548248	438898128 (82.41)
11	538133304	443547942 (82.42)
10	533043760	442286658 (82.97)
9	487157520	384761774 (78.98)
8	579071400	487448476 (84.18)
7	635878548	512018012 (80.52)
6	680312528	571790298 (84.05)
5	725057768	598995030 (82.61)
4	759008900	636939292 (83.92)
3	792397628	663182452 (83.69)
2	962189580	801385614 (83.29)
1	921909280	738431768 (80.10)

Table 4.3: 27-mers table. For each chromosome given the numbers of total and unique 27-mers (with the percentage from the total number). The 27-mers were counted in the context of 4 transformations of chromosome sequence.

chromosome	total 29-mers	unique 29-mers (% from total)
Y	105653628	72875152 (68.97)
X	619568596	494838776 (79.87)
22	156633732	122810180 (78.41)
21	160348764	130738170 (81.53)
20	255767060	215127218 (84.11)
19	233762136	171958096 (73.56)
18	320353236	269548768 (84.14)
17	331675600	257341282 (77.59)
16	327221420	260634674 (79.65)
15	338563380	275341372 (81.33)
14	362269908	308529916 (85.17)
13	391930144	343867552 (87.74)
12	532548016	448276208 (84.18)
11	538133176	452658720 (84.12)
10	533043152	451138410 (84.63)
9	487157160	392271140 (80.52)
8	579071312	497112280 (85.85)
7	635878396	523201220 (82.28)
6	680312408	583818488 (85.82)
5	725057480	611687370 (84.36)
4	759008764	650697038 (85.73)
3	792397404	677661600 (85.52)
2	962189340	819340808 (85.15)
1	921908256	756166828 (82.02)

Table 4.4: 29-mers table. For each chromosome given the numbers of total and unique 29-mers (with the percentage from the total number). The 29-mers were counted in the context of 4 transformations of chromosome sequence.

Histograms of ± 50-windows coverages (chromosome 6 example)

With the chromosome 6 as an example, we aim to give an idea about how many CpGs have how many unique k-mers in different ranges of proximity to them (we dive into cases of within 10, 20, 50 and 100 basepairs, Fig. 4.1).

4.3 Performance

In the following table (Table 4.5) are presented the observations regarding the times needed at the preparational phase (creation of the storing, searching, and counter structures; performed once at the beginning of the run) for each chromosome, and suggestions about the number of exploited processes at the multiprocessing stage. The suggestions are made for the program's execution on a machine with 16 cores (32 virtual ones) and 128GB RAM available.

File processing times

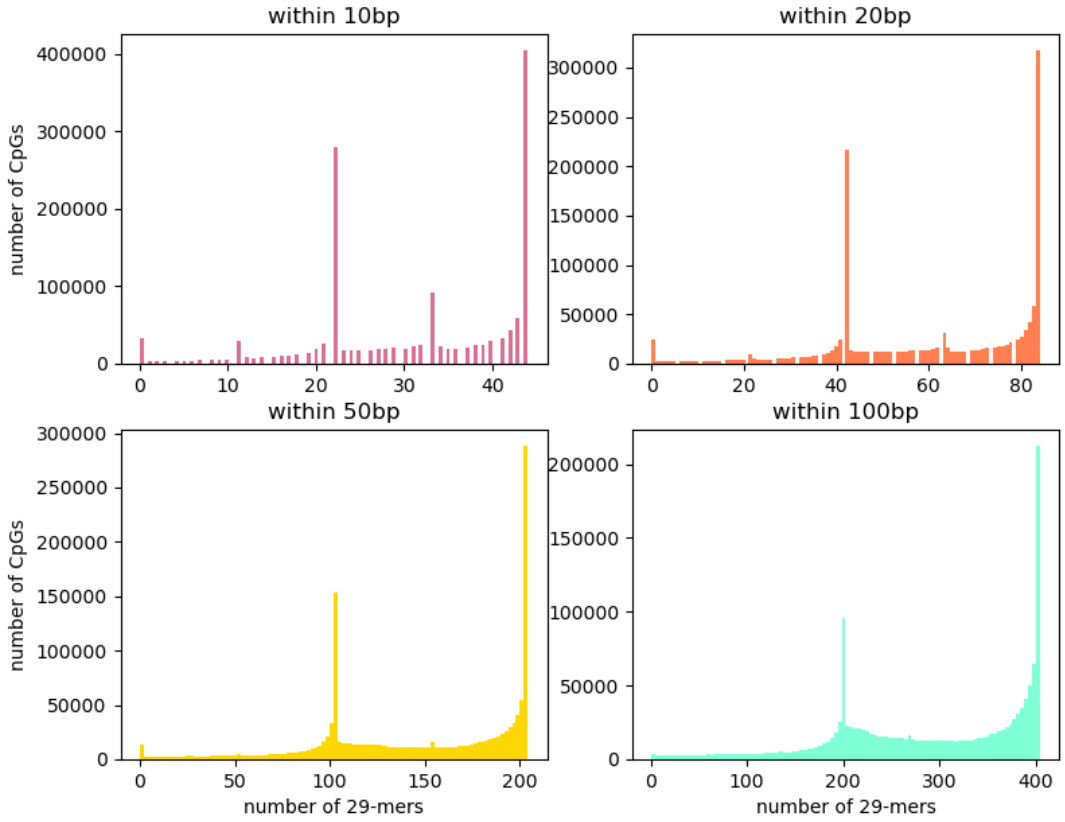

Figure 4.1: Histograms of the amounts of "marker" 29-mers in the ranges of 10, 20, 50 and 100 basepairs around the CpG positions (chromosome 6).

chromosome	preparational phase	recommended № of processes
Y	≈ 7min	10
X	≈ 1h25min	7
22	≈ 30min	10
21	≈ 25min	10
20	≈ 50min	10
19	≈ 50min	10
18	≈ 55min	10
17	≈ 1h10min	7
16	≈ 1h5min	7
15	≈ 1h	7
14	≈ 1h	7
13	≈ 1h5min	7
12	≈ 1h30min	5
11	≈ 1h35min	5
10	≈ 1h40min	3
9	≈ 1h20min	5
8	≈ 1h40min	3
7	≈ 1h50min	2
6	≈ 1h55min	2
5	≈ 2h10min	2
4	≈ 2h5min	2
3	≈ 2h20min	1
2	≈ 3h	1
1	≈ 3h15min	1

Table 4.5: Current performance of the method. For each chromosome is given the approximate time of the preparational phase and recommended number of processes for the multiprocessing stage under the assumption of 128GB RAM and 16 cores available on the working machine.

Based on the retrieved processing times for the sample files with known size and the number of processes used for parallelized calculations, the approximate processing speed stands at $\approx$ 0.27 Mbp/s (= 16.2 Mbp/min).

Calculation of total working time

Considering the notes above, the total working time of one run of the tool (which will perform methylation calling on a sample directory k in the context of a particular chromosome c) can be calculated as follows:

working time of the run in minutes $\approx$ preparational phase of the chromosome c (in minutes) + $\sum_{foreachfilefink}$ (size of f in Mbp / number of processes used during multiprocessing) / (16.2)

In the case of multiple runs, the overall working time is inferred by summing the working times of all included runs.

4.4 Results comparison

Here we would like to present a comparison of results from wg-blimp and our software, evaluated on two WGBS data samples for chromosome 6.

4.4.1 Counts difference comparison

This subsection contains several visualizations of the differences found between the results obtained from our tool and wg-blimp.

Sum of absolute difference values

Histograms in Figures 4.2 and 4.3 show us a distribution of sums of absolute difference values, defined as follows:

| methylated counter in our file - methylated counter in wg-blimp file | +

| unmethylated counter in our file - unmethylated counter in wg-blimp file | .

All the values $\geqslant$ 100 were counted together and, among others, contain the outlier CpGs, which were most probably mistakenly counted due to different mutations (for a more detailed description of the matter look to Section 5.2.2).

Even though values higher or equal to 100 are the most frequent, overall those values account for a rather small portion of all CpG positions that we have data on. If we exclude those, we can observe another peak at $\approx$ 15.

Differences in methylated counts

Histograms in Figures 4.4 and 4.5 depict the differences in methylated

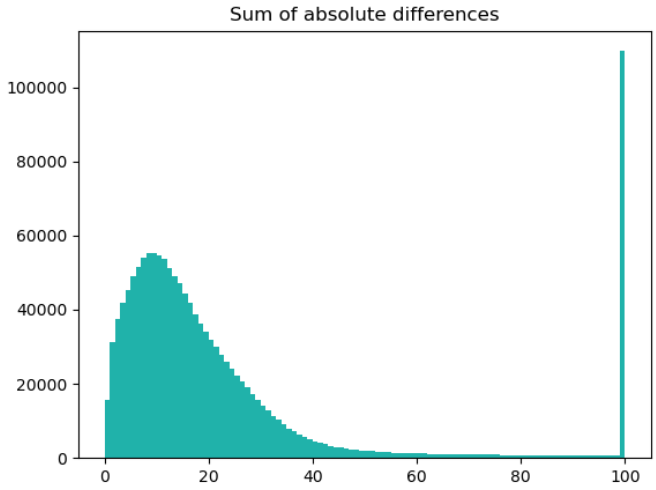

Figure 4.2: Histogram of the sum of absolute difference values of methylation calling results obtained from our tool and wg-blimp on Blood-Macrophages WGBS data (Sample_43_Hm03_BlMa_Ct2_WGBS_E_TS) for chromosome 6. X-axis: | methylated counter in our file - methylated counter in wg-blimp file | + | unmethylated counter in our file - unmethylated counter in wg-blimp file | ; y-axis: amount of CpG positions, having the corresponding x-axis value.

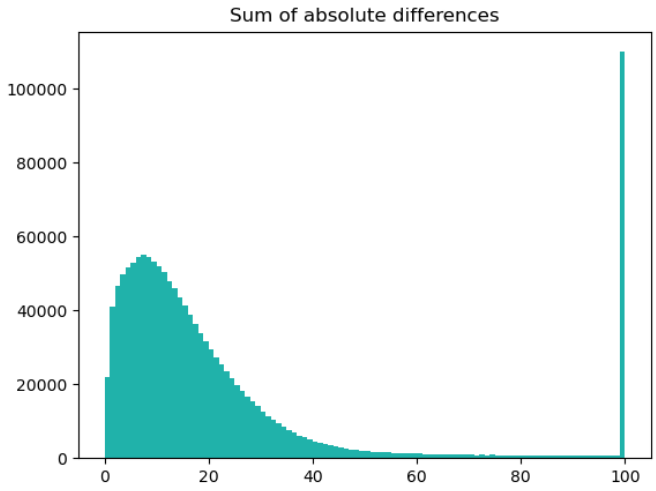

Figure 4.3: Histogram of the sum of absolute difference values of methylation calling results obtained from our tool and wg-blimp on Blood-Monocytes WGBS data (Sample_43_Hm03_BlMo_Ct2_WGBS_E_TS) for chromosome 6. X-axis: | methylated counter in our file - methylated counter in wg-blimp file | + | unmethylated counter in our file - unmethylated counter in wg-blimp file | ; y-axis: amount of CpG positions, having the corresponding x-axis value.

counters following the formula: methylated counter in our file - methylated counter in wg-blimp file.

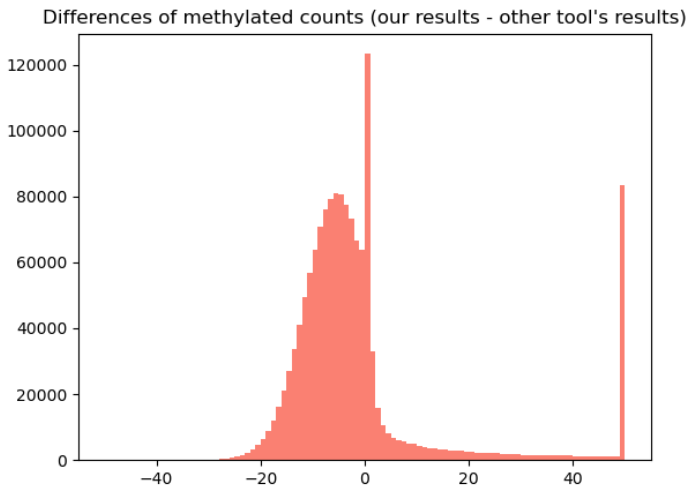

Figure 4.4: Histogram of the differences in methylated counts of methylation calling results obtained from our tool and wg-blimp on Blood-Macrophages WGBS data (Sample_43_Hm03_BlMa_Ct2_WGBS_E_TS) for chromosome 6. X-axis: methylated counter in our file - methylated counter in wg-blimp file; y-axis: amount of CpG positions, having the corresponding x-axis value.

All the values $\geqslant 50$ were counted together and $\leqslant$ -50 were counted together as well.

Apart from higher counts of values $\geqslant 50$, here are very prominent peaks at 0 and $\approx$ -7. Overall for a particular CpG, our tool tends to declare less of its occurrences methylated, than wg-blimp.

Differences in unmethylated counts

Histograms in Figures 4.6 and 4.7 depict the differences in unmethylated counters following the formula: unmethylated counter in our file - unmethylated counter in wg-blimp file.

All the values $\geqslant 50$ were counted together and $\leqslant$ -50 were counted together as well.

For the unmethylated counters, we can observe an opposite to the one for the methylated counters situation: our tool seems to assume more occurrences of a CpG unmethylated, than wg-blimp. Regarding the peaks, only a striking one at 0 is present, apart from the one for values over 50.

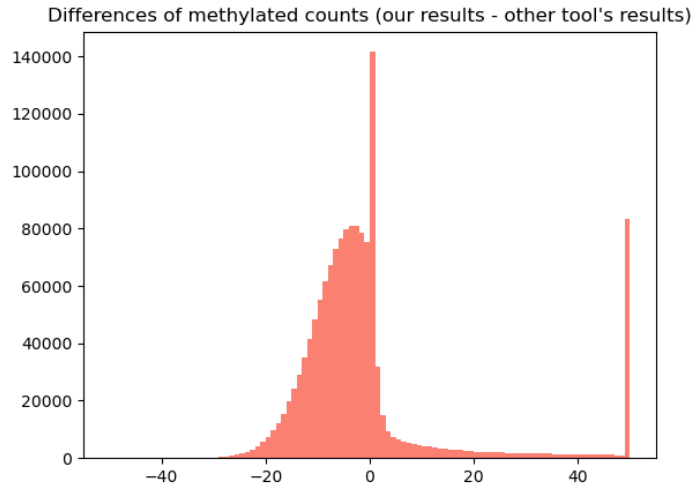

Figure 4.5: Histogram of the differences in methylated counts of methylation calling results obtained from our tool and wg-blimp on Blood-Monocytes WGBS data (Sample_43_Hm03_BlMo_Ct2_WGBS_E_TS) for chromosome 6. X-axis: methylated counter in our file - methylated counter in wg-blimp file; y-axis: amount of CpG positions, having the corresponding x-axis value.

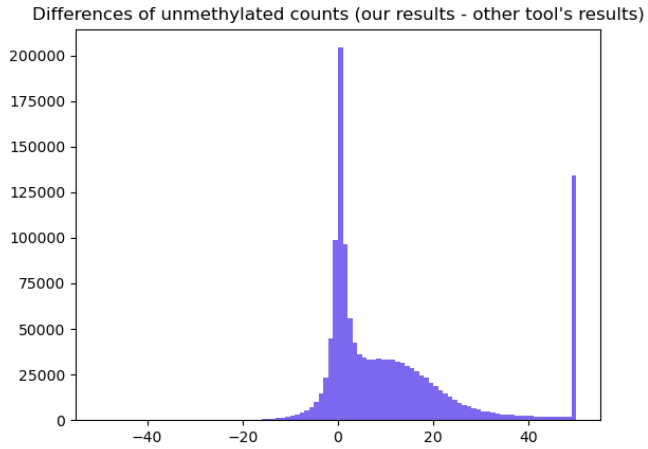

Figure 4.6: Histogram of the differences in unmethylated counts of methylation calling results obtained from our tool and wg-blimp on Blood-Macrophages WGBS data (Sample_43_Hm03_BlMa_Ct2_WGBS_E_TS) for chromosome 6. X-axis: unmethylated counter in our file - unmethylated counter in wg-blimp file; y-axis: amount of CpG positions, having the corresponding x-axis value.

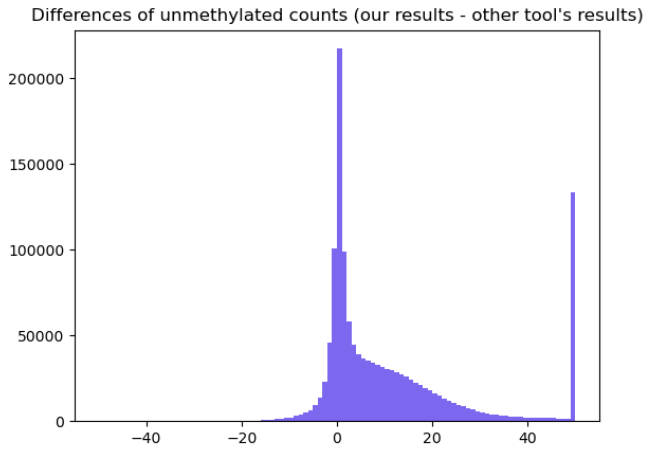

Figure 4.7: Histogram of the differences in unmethylated counts of methylation calling results obtained from our tool and wg-blimp on Blood-Monocytes WGBS data (Sample_43_Hm03_BlMo_Ct2_WGBS_E_TS) for chromosome 6. X-axis: unmethylated counter in our file - unmethylated counter in wg-blimp file; y-axis: amount of CpG positions, having the corresponding x-axis value.

2D-histogram of differences

Figures 4.8 and 4.9 represent cumulative 2-dimensional histograms of the abovementioned counter differences.

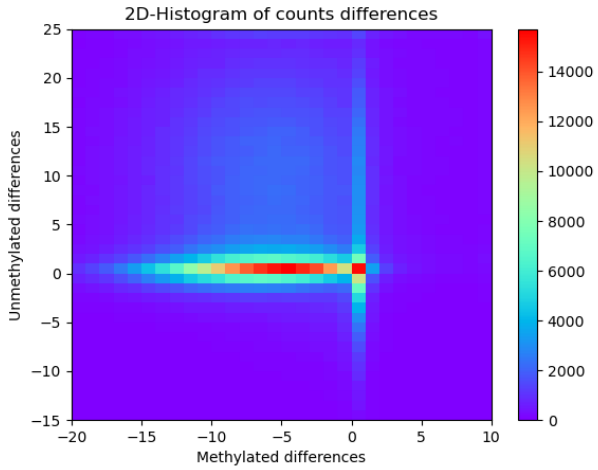

Figure 4.8: 2D-histogram of the differences in counts of methylation calling results obtained from our tool and wg-blimp on Blood-Macrophages WGBS data (Sample_43_Hm03_BlMa_Ct2_WGBS_E_TS) for chromosome 6. X-axis: methylated counter in our file - methylated counter in wg-blimp file; y-axis: unmethylated counter in our file - unmethylated counter in wg-blimp file.

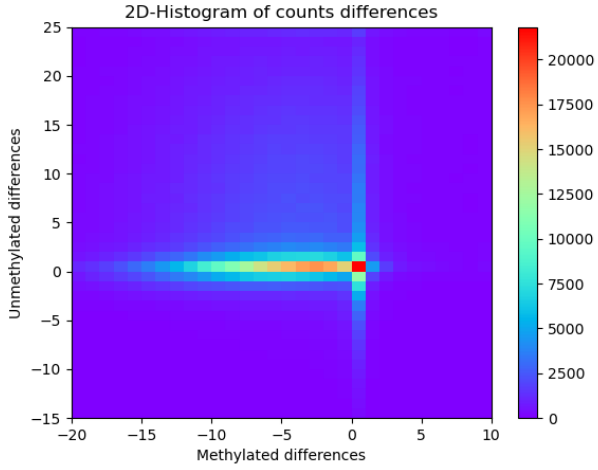

Figure 4.9: 2D-histogram of the differences in counts of methylation calling results obtained from our tool and wg-blimp on Blood-Monocytes WGBS data (Sample_43_Hm03_BlMo_Ct2_WGBS_E_TS) for chromosome 6. X-axis: methylated counter in our file - methylated counter in wg-blimp file; y-axis: unmethylated counter in our file - unmethylated counter in wg-blimp file.

The 2-dimensional histograms give us a summarized overview of the differences between the results of the tools. With the fully matching results being consistently the highest peaks, however, we can also observe 3 most prevalent cases:

1. Unmethylated difference close to 0 together with a negative methylated difference (from 0 to $\approx$ -20),
2. Methylated difference at 0 together with unmethylated difference going in both directions, but further on the positive axis,
3. Negative methylated difference (from 0 to $\approx$ -13) together with the positive unmethylated difference (from 0 to $\approx$ 20).

Out of these plots, we can conclude that even though our tool appears to slightly lower the estimated methylation level as compared with wg-blimp, the results obtained from both tools are still relatively close.

4.4.2 Presence of CpGs

In this subsection, we provide some numbers regarding the presence of data on CpGs in the resulting file of our tool and wg-blimp.

Total number of CpGs in chromosome 6: 1511189.

Number of CpGs in chromosome 6 that can be identified with our tool: 1501775 (99.38% from total).

Number of CpGs in chromosome 6 that can not be identified with our tool: 9414 (0.62% from total).

In Table 4.6 we provide the CpGs presence numbers related to the analyzed WGBS samples.

sample	only in our tool	a	only in wg-blimp	b	not present
Macrophages	30144 (1.99%)	69780 (4.62%)	63532 (4.20%)	7263 (0.48%)	2151 (0.14%)
Monocytes	35288 (2.33%)	81611 (5.40%)	71843 (4.75%)	7188 (0.48%)	2226 (0.15%)

Table 4.6: Presence of CpGs, sample-wise. All the percentage values are given from the total number of CpGs in chromosome 6. Columns decryption: "only in our tool" - CpGs seen only in resulting files from our tool; "a" - CpGs that can be identified with our methodology, but are not seen by our tool in the sample reads; "only in wg-blimp" - CpGs seen only in resulting files from wg-blimp; "b" - CpGs that cannot be identified by our tool and are recognized by wg-blimp; "not present" - CpGs that are not seen in the samples by both of the tools.

From the table, we can infer that the combination of the tools' results minimizes the number of completely unseen CpGs. Both the tools can offer results on the CpGs that are seen only by this tool, as well as tools

like wg-blimp can probably cover the losses in our tool's results caused by k-mer disrupting mutations.

Chapter 5
Conclusion

Towards the end, I would like to highlight some key properties of the described method, as well as discuss noticed drawbacks and possible ways for improvement.

5.1 Summary

In the course of this work was proposed an approach that adopts the utilization of unique k-mers for performing alignment-free methylation calling on WGBS data. Based on this methodology software produces comparable methylation calling results to those, obtained from pipelines like wg-blimp, in a reasonable time. At the current state, our tool might not necessarily outperform the existing tools for the analysis of WGBS data, but with the implementation of upgrading ideas, it may show a great deal of potential.

5.2 Discussions

5.2.1 Advantages

The main goal of the method was to avoid the alignment step in the methylation calling procedure to save computational resources.

Since processing times for one file of the sample remain nearly constant regardless of the chromosome size and preparational stage times for each chromosome are also known, the overall working time can be calculated

beforehand based on the set of chromosomes in interest, amount of processes used at the multiprocessing stage and the sizes of files in the sample directory.

Usage of several processes for file handling allows to further scale down the base processing time for 1 process by the number of processes.

Tool's structure allows it to be updated for new releases of the human reference genome sequence just by reconstruction of the lists of CpG positions and lists of "marker" 29-mers.

The output file format allows to further develop and perform custom analysis steps for the user's specific needs.

5.2.2 Disadvantages

Though the vast majority of the CpG positions can be identified with the "marker" 29-mers, there is still a portion of CpGs that are invisible for the tool.

Usage of the tool's main script requires the CpG positions lists and "marker" 29-mers lists for chromosomes of interest to be constructed beforehand.

Since RAM usage grows with the rise of the number of processes utilized at the multiprocessing stage, the overall processing time depends on the user's technical resources (namely on the number of cores and RAM available).

At the moment proposed methodology is sensitive to the indel mutations inside the ± 50-window because they create an additional offset to the shift value, which is currently not accounted for. This means that in case of an indel mutation inside the ± 50-window of a particular CpG, the position calculated by summing the starting index of the found "marker" 29-mer and its shift value is not the associated CpG position inside the read, if the indel mutation occurred at the same half of the window, as this "marker" 29-mer.

Moreover, some mutations can lead to forming such 29-mers in the read, that their CtoT or GtoA transformation coincides with some "marker" 29-mer. Such situations will result in irregular numbers, since regions with mutations will be mistakenly recognized as the "marker" 29-mers and results will be collected along with the ones for the true CpG occurrences. The probability for such confusions to appear for a CpG position (as well as their impact) increases with the number of associated with it "marker" 29-mers.

5.2.3 Ideas

In case this or any future version of this tool becomes publicly available, storing already generated for the current human reference genome version lists of CpG positions and lists of "marker" 29-mers along with the scripts for their construction would allow the user to either download only the files for desired chromosomes, or to generate their own in case they need to analyze the data in the context of any other reference genome.

The problem of indel mutations inside ± 50-window could partially be solved by double checking, whether a CpG position inside the read, estimated by a "marker" 29-mer from one half of the window, corresponds to the one, estimated by a "marker" 29-mer from the other half. Unfortunately, such a checking procedure requires at least one "marker" 29-mer from each side of the window to exist and be present in the read, which is not always possible.

Usage of different masks of gapped 29-mers and their combinations could provide more robust and accurate results by increasing the coverage of visible to the method CpG positions up to 100%.

To address the problem of mutation-caused false occurrences of the "marker" 29-mers the usage of strongly unique instead of just unique 29-mers could be applied. In addition, the lists of "marker" 29-mers could be revised to find some reasonable upper limit for the amount of the "marker" 29-mers per a CpG position, and the lists could be filtered adequately to leave only the closest and most significant to their CpGs "marker" 29-mers and reduce the abundance of 29-mers associated with many CpG positions.

Chapter 6
Appendix

6.1 Open source link

All the scripts created and used during this work can be found at `https://github.com/oliadikan/thesis_source`.

6.2 Inputs generation example

Beginning with the reference genome file at /home/olyadikan/Documents/Thesis/GRCh38_latest_genomic.fna .

1. python3 1_chrom_separation_upper_th.py /home/olyadikan/Documents/Thesis/GRCh38_latest_genomic.fna /home/olyadikan/Documents/Thesis/chroms ; Result: ChrN.fasta files at /home/olyadikan/-Documents/Thesis/chroms, N - chromosome number.

2. python3 2_cpgs_th.py < /home/olyadikan/Documents/Thesis/chroms/ChrN.fasta > /home/olyadikan/Documents/Thesis/cpgs/CpGs_N.txt ; N - chromosome number, executing for each chromosome.

3. python3 3_direct_transforms_th.py /home/olyadikan/Documents/Thesis/chroms/Chr /home/olyadikan/Documents/Thesis/direct ; Result: ChrN_CtoT.fasta and ChrN_GtoA.fasta files at /home/olyadikan/-Documents/Thesis/direct, N - chromosome number.

4. seqkit seq -r -p /home/olyadikan/Documents/Thesis/direct/ChrN_CtoT.fasta > /home/olyadikan/Documents/Thesis/reverse/ChrN_CtoT_rev.fasta -t dna ; N - chromosome number, executing for each chromosome for both CtoT and GtoA transformations.
5. kmc -k29 -fm -ci1 -cx1 -b /home/olyadikan/Documents/Thesis/kmc/kmc3_input.fasta /home/olyadikan/Documents/Thesis/kmc/29mers.res /home/olyadikan/Documents/Thesis/kmc/kmc_tmp_dir/ ; kmc3_input.fasta contains all the ChrN_CtoT.fasta, ChrN_GtoA.fasta, ChrN_CtoT_rev.fasta, and ChrN_GtoA_rev.fasta sequences.
6. ./kmc_tools transform /home/olyadikan/Documents/Thesis/kmc/29mers.res -ci1 -cx1 dump /home/olyadikan/Documents/Thesis/unique_29mers.txt
7. python3 4_separating_uniques_th.py /home/olyadikan/Documents/Thesis/unique_29mers.txt /home/olyadikan/Documents/Thesis ; Result: unique_29mers_ctot, unique_29mers_gtoa, and unique_29mers_at at /home/olyadikan/Documents/Thesis .
8. python3 5_un29_chrom_division_th.py /home/olyadikan/Documents/Thesis/direct/Chr /home/olyadikan/Documents/Thesis/reverse/Chr /home/olyadikan/Documents/Thesis /home/olyadikan/Documents/Thesis/uniques ; Result: unique_29mers_chrN_ctot, unique_29mers_chrN_gtoa, unique_29mers_chrN_ctot_rev, and unique_29mers_chrN_gtoa_rev files (N - chromosome number) for all chromosomes at /home/olyadikan/Documents/Thesis/uniques .
9. python3 6_unique_per_cpg_lists_th.py /home/olyadikan/Documents/Thesis/direct/Chr /home/olyadikan/Documents/Thesis/reverse/Chr /home/olyadikan/Documents/Thesis/cpgs/CpGs_ /home/olyadikan/Documents/Thesis/uniques/unique_29mers_chr /home/olyadikan/Documents/Thesis/lists ; Result: N_cpgs_kmers_list_upd and N_cpgs_kmers_list_rev files (N - chromosome number) for all chromosomes at /home/olyadikan/Documents/Thesis/lists

After this procedure the files that we need for our method are at /home/olyadikan/Documents/Thesis/cpgs and /home/olyadikan/Documents/Thesis/lists .

6.3 Arguments of the main script

For more convenient usage the main working script of the method method_rupd_wod_snmk_th.py is incorporated in a Snakemake workflow with 2 modes of work. However, if needed, it can also be executed on its own. Here is the explanation of the parameters:

- '-s', '–sample': path to the directory with samples subdirectories, is required.
- '-d', '–directory': name of the directory containing sample files, is required.
- '-c', '–chrom': chromosome number (or X/Y) for the methylation state calling, is required.
- '-p', '–process': number of processes used at the multiprocessing stage, default value is 5. Recommended values for each chromosome are given in modus_1.sh .
- '-l', '–log': path for the log file, is required.
- '-i', '–lists': path to the directory with the "marker" 29-mers lists, is required.
- '-k', '–cpgs': path to the file, containing CpG positions (up to N.txt, N - number of chromosome), is required.
- '-o', '–output': path for the output file, is required.

6.4 Snakefile configuration example

```
rule modus_1:
 # Processing one sample for multiple chromosomes.
 # md1_dir and output have to be set up.
 # The set of processed chromosomes is controlled by commenting/uncommenting dedicated lines in
 # modus_1.sh.
 input:
  lists = "/home/olyadikan/Documents/Thesis/lists", # path to the directory with "marker" unique 29
  cpgs = "/home/olyadikan/Documents/Thesis/cpgs/CpGs_", # path to CpG positions files (up to N.txt,
  samples = "/home/olyadikan/Documents/Thesis/wgbs-deep-moma/raw" # path to the directory with samp
 params:
  drt = config['dir'], # sample directory
  md1_dir = "/home/olyadikan/Documents/Thesis/snmk/modus_1", # modus directory
  out = "".join([md1_dir, "/", config['dir']]), # output directory
  out_l = "".join([md1_dir, "/", config['dir'], "/log"]) # directory for log files
 output:
  "/home/olyadikan/Documents/Thesis/snmk/finished_m1.txt" # file, existence of which signals to the
  #that job succeded (file MUST NOT exist beforehand, directory MUST exist beforehand!)
 script:
  "modus_1.sh"
```

Figure 6.1: Example of Snakefile configuration for modus_1.

```
rule modus_2:
 # Processing multiple samples for one chromosome.
 # drt, md2_dir and output have to be set up.
 # The set of processed samples is controlled by commenting/uncommenting (adding if necessary)
 # dedicated lines in modus_2.sh.
 input:
  lists = "/home/olyadikan/Documents/Thesis/lists", # path to the directory with "marker" unique 29
  cpgs = "/home/olyadikan/Documents/Thesis/cpgs/CpGs_", # path to CpG positions files (up to N.txt,
  samples = "/home/olyadikan/Documents/Thesis/wgbs-deep-moma/raw" # path to the directory with samp
 params:
  drt = ["Sample_43_Hm03_BlMa_Ct2_WGBS_E_TS", "Sample_43_Hm03_BlMo_Ct2_WGBS_E_TS"], # sample direct
  chrom = config['chrom'], # chromosome number
  pr = config['proc'], # amount of processes
  md2_dir = "/home/olyadikan/Documents/Thesis/snmk/modus_2", # modus directory
  l = "".join([md2_dir, "/log"]) # directory for log files
 output:
  "/home/olyadikan/Documents/Thesis/snmk/finished_m2.txt" # file, existence of which signals to the
  #that job succeded (file MUST NOT exist beforehand, directory MUST exist beforehand!)
 script:
  "modus_2.sh"
```

Figure 6.2: Example of Snakefile configuration for modus_2.

6.5 Snakemake command example

snakemake modus_1 –config dir="Sample_43_Hm03_BlMa_Ct2_WGBS_E_EC" chrom=6 proc=2 –cores all *(Works with rule modus_2 as well.)*

6.6 Main script's execution command example

python3 method_rupd_wod_snmk_th.py -d Sample_43_Hm03_BlMa_Ct2_WGBS_E_EC -c 6 -p 2 -l /home/olyadikan/Documents/Thesis/snmk/-modus_1/Sample_43 _Hm03_BlMa_Ct2_WGBS_E_EC/log -i /home/olyadikan/-Documents/Thesis/lists -k /home/olyadikan/Documents/Thesis/cpgs/CpGs_ -o /home/olyadikan/Documents/Thesis/snmk/modus_1/Sample_43_Hm03 _BlMa_Ct2_WGBS_E_EC -s /home/olyadikan/Documents/Thesis/wgbs-deep-moma/raw

6.7 File structure of the given example

For the given example, after the generation of input files and the execution of the given Snakemake command for modus_1 and modus_2 with the provided Snakefile configuration, we obtain the following file structure:

```
-/home/olyadikan/Documents/Thesis
.../chroms
......-/ChrN.fasta
.../direct
......-/ChrN_CtoT.fasta
......-/ChrN_GtoA.fasta
.../reverse
......-/ChrN_CtoT_rev.fasta
......-/ChrN_GtoA_rev.fasta
.../kmc
......-/kmc_tmp_dir (empty after kmc execution)
......-/kmc3_input.fasta
......-/29mers.res
.../uniques
......-/unique_29mers_chrN_ctot
......-/unique_29mers_chrN_gtoa
......-/unique_29mers_chrN_ctot_rev
......-/unique_29mers_chrN_gtoa_rev
.../lists
......-/N_cpgs_kmers_list_upd
......-/N_cpgs_kmers_list_rev
.../cpgs
......-/CpGs_N.txt
.../snmk
......-/modus_1
.........-/Sample_43_Hm03_BlMa_Ct2_WGBS_E_EC
............-/log
..............-/log files
............-/results for each chromosome of interest
......-/modus_2
.........-/log
............-/log files
.........-/results for Sample_43_Hm03_BlMa_Ct2_WGBS_E_TS and Sample_43_Hm03_BlMo_Ct2_WGBS_E_TS (chrom 6)
......-/finished_m1.txt
......-/finished_m2.txt
.../wgbs-deep-moma
......-/raw
.........-/Sample_43_Hm03_BlMa_Ct2_WGBS_E_EC
............-/sample files
.........-/Sample_43_Hm03_BlMa_Ct2_WGBS_E_TS
............-/sample files
.........-/Sample_43_Hm03_BlMo_Ct2_WGBS_E_TS
............-/sample files
.../GRCh38_latest_genomic.fna
.../unique_29mers.txt
.../unique_29mers_ctot
.../unique_29mers_gtoa
.../unique_29mers_at
```

Figure 6.3: File structure of the example

6.8 log file

Here is presented a log file, produced in the course of main script's execution, with explanation.

Sample_43_Hm03_BlMa_Ct2_WGBS_E_TS *(name of sample's directory)*

6 2023-01-03 20:07:05.508106 *(chromosome number, timestamp)*

/scratch/wgbs-deep-moma/raw/Sample_43_Hm03_BlMa_Ct2_WGBS_E_TS/43_Hm03_BlMa_Ct2_WGBS_E_TS_TGACCA_L005_R1_001.fastq.gz *(currently processed sample file (start of block))*

start 2023-01-03 21:59:35.764364 *(start of file reading, timestamp)*

read 2023-01-03 21:59:59.539968 *(file read, timestamp)*

cores 2 *(number of processes used at the multiprocessing stage)*

mp 2023-01-03 22:00:00.054415 *(start of multiprocessing, timestamp)*

processed 2023-01-03 22:45:27.934875 *(end of multiprocessing (file processed), timestamp (end of block))*

(This block repeats for each file in /scratch/wgbs-deep-moma/raw/Sample_43_Hm03_BlMa_Ct2_WGBS_E_TS).

directory done 2023-01-05 09:18:23.214134 *(all files in the directory processed, timestamp)*

finish 2023-01-05 09:18:28.942016 *(script's execution finished successfully, timestamp)*

Bibliography

Felix Krueger and Simon Andrews. Bismark: a flexible aligner and methylation caller for Bisulfite-Seq applications. *Bioinformatics*, 27(11):1571–1572, 04 2011. ISSN 1367-4803. URL https://doi.org/10.1093/bioinformatics/btr167.

Weilong Guo, Petko Fiziev, Weihong Yan, et al. BS-Seeker2: a versatile aligning pipeline for bisulfite sequencing data. *BMC Genomics*, 14(774), 2013. URL https://doi.org/10.1186/1471-2164-14-774.

Benjamin Kreck, George Marnellos, Julia Richter, Felix Krueger, Reiner Siebert, and Andre Franke. B-SOLANA: an approach for the analysis of two-base encoding bisulfite sequencing data. *Bioinformatics*, 28(3): 428–429, 12 2011. ISSN 1367-4803. URL https://doi.org/10.1093/bioinformatics/btr660.

Devon Ryan and Dan Ehninger. Bison: bisulfite alignment on nodes of a cluster. *BMC Bioinformatics*, 15(337), 2014. URL https://doi.org/10.1186/1471-2105-15-337.

Changjin Hong, Nathan Clement, Spencer Clement, et al. Probabilistic alignment leads to improved accuracy and read coverage for bisulfite sequencing data. *BMC Bioinformatics*, 14(337), 2013. URL https://doi.org/10.1186/1471-2105-14-337.

Joungmin Choi, Yoonjae Park, Sun Kim, and Heejoon Chae. Cloud-BS: A MapReduce-based bisulfite sequencing aligner on Cloud. *Journal of Bioinformatics and Computational Biology*, 16(6), 10 2018. URL https://doi.org/10.1142/S0219720018400280.

Karen Wong, Walfred Ma, Chun-Yu Wei, et al. Towards a reference genome that captures global genetic diversity. *Nature Communications*, 11(5482), 2020. URL https://doi.org/10.1038/s41467-020-19311-w.

Sergey Nurk, Sergey Koren, Arang Rhie, Mikko Rautiainen, et al. The complete sequence of a human genome. *Science*, 376(6588):44–53, 2022. URL https://www.science.org/doi/abs/10.1126/science.abj6987.

International Human Genome Sequencing Consortium. Initial sequencing and analysis of the human genome. *Nature*, 409:860–921, 2001. URL https://doi.org/10.1038/35057062.

Valerie Schneider, Tina Graves-Lindsay, Kerstin Howe, et al. Evaluation of GRCh38 and de novo haploid genome assemblies demonstrates the enduring quality of the reference assembly. *Genome Research*, 27:849–864, 2017. URL https://doi.org/10.1101/gr.213611.116.

Genome Reference Consortium. Ideogram of the latest human assembly, GRCh38.p14, 2022. URL https://www.ncbi.nlm.nih.gov/grc/human. [Online; accessed December 25, 2022].

Lisa Moore, Thuc Le, and Guoping Fan. DNA Methylation and Its Basic Function. *Neuropsychopharmacology*, 38:23–38, 2013. URL https://doi.org/10.1038/npp.2012.112.

Dong-Qiao Shi, Iftikhar Ali, Jun Tang, and Wei-Cai Yang. New Insights into 5hmC DNA Modification: Generation, Distribution and Function. *Frontiers in Genetics*, 8(100), 2017. URL https://doi.org/10.3389/fgene.2017.00100.

Bilian Jin, Yajun Li, and Keith Robertson. DNA methylation: superior or subordinate in the epigenetic hierarchy? *Genes & cancer*, 2 6:607–17, 2011. URL https://doi.org/10.1177/1947601910393957.

Serge Saxonov, Paul Berg, and Douglas Brutlag. A genome-wide analysis of CpG dinucleotides in the human genome distinguishes two distinct classes of promoters. *Proceedings of the National Academy of Sciences of the United States of America*, 103 5:1412–1417, 2006. URL https://doi.org/10.1073/pnas.0510310103.

Venkataramana Kandi and Sabitha Vadakedath. Effect of DNA Methylation in Various Diseases and the Probable Protective Role of Nutrition: A Mini-Review. *Cureus*, 7 8:e309, Aug 2015. URL https://doi.org/10.7759/cureus.309.

Zelin Jin and Yun Liu. DNA methylation in human diseases. *Genes & Diseases*, 5:1–8, 2018. URL https://doi.org/10.1016/j.gendis.2018.01.002.

Oliver McDonald, Xin Li, Tyler Saunders, et al. Epigenomic reprogramming during pancreatic cancer progression links anabolic glucose metabolism to distant metastasis. *Nature Genetics*, 49 3:367–376, 2017. URL https://doi.org/10.1038/ng.3753.

M Frommer et al. A genomic sequencing protocol that yields a positive display of 5-methylcytosine residues in individual DNA strands. *Proceedings of the National Academy of Sciences of the United States of America*, 89 5:1827–1831, 1992. URL https://doi.org/10.1073/pnas.89.5.1827.

Alan Harrison and Anne Parle-McDermott. DNA methylation: a timeline of methods and applications. *Frontiers in Genetics*, 2(74), 2011. URL https://doi.org/10.3389/fgene.2011.00074.

Jill Adams. Complex genomes: Shotgun sequencing. *Nature Education*, 1 1 (186), 2008.

Eric Green. Strategies for the systematic sequencing of complex genomes. *Nature Reviews Genetics*, 2:573–583, 2001. URL https://doi.org/10.1038/35084503.

Yuanyuan Li and Trygve Tollefsbol. *DNA Methylation Detection: Bisulfite Genomic Sequencing Analysis*, pages 11–21. Humana Press, Totowa, NJ, 2011. ISBN 978-1-61779-316-5. URL https://doi.org/10.1007/978-1-61779-316-5_2.

Ruiqiang Li, Hongmei Zhu, Jue Ruan, et al. De novo assembly of human genomes with massively parallel short read sequencing. *Genome Research*, 20 2:265–272, 2010. URL https://doi.org/10.1101/gr.097261.109.

Alkes Price, Neil Jones, and Pavel Pevzner. De novo identification of repeat families in large genomes. *Bioinformatics*, 21(suppl_1):i351–i358, 06 2005. ISSN 1367-4803. URL https://doi.org/10.1093/bioinformatics/bti1018.

Prabina Kumar Meher, Tanmaya Kumar Sahu, and A.R. Rao. Identification of species based on DNA barcode using k-mer feature vector and Random forest classifier. *Gene*, 592(2):316–324, 2016. ISSN 0378-1119. URL https://doi.org/10.1016/j.gene.2016.07.010.

Karl Nordström, Maria Albani, Geo James, et al. Mutation identification by direct comparison of whole-genome sequencing data from mutant and wild-type individuals using k-mers. *Nature Biotechnology*, 31:325–330, 2013. URL https://doi.org/10.1038/nbt.2515.

Ying Wang et al. Identifying Group-Specific Sequences for Microbial Communities Using Long k-mer Sequence Signatures. *Frontiers in Microbiology*, 9(872), 2018. URL https://doi.org/10.3389/fmicb.2018.00872.

Guillaume Marçais and Carl Kingsford. A fast, lock-free approach for efficient parallel counting of occurrences of k-mers. *Bioinformatics*, 27 (6):764–770, 01 2011. ISSN 1367-4803. URL https://doi.org/10.1093/bioinformatics/btr011.

Marek Kokot, Maciej Długosz, and Sebastian Deorowicz. KMC 3: counting and manipulating k-mer statistics. *Bioinformatics*, 33(17):2759–2761, 05 2017. ISSN 1367-4803. URL https://doi.org/10.1093/bioinformatics/btx304.

Marius Erbert, Steffen Rechner, and Matthias Müller-Hannemann. Gerbil: a fast and memory-efficient k-mer counter with GPU-support. *Algorithms for Molecular Biology*, 12(9), 2017. URL https://doi.org/10.1186/s13015-017-0097-9.

Swati Manekar and Shailesh Sathe. A benchmark study of k-mer counting methods for high-throughput sequencing. *GigaScience*, 7(12), 10 2018. ISSN 2047-217X. URL https://doi.org/10.1093/gigascience/giy125. giy125.

Ben Langmead, Cole Trapnell, Mihai Pop, and Steven Salzberg. Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome Biology*, 10(R25), 2009. URL https://doi.org/10.1186/gb-2009-10-3-r25.

Michael Burrows and David Wheeler. A Block Sorting Lossless Data Compression Algorithm. Technical Report 124. *Palo Alto, CA: Digital Equipment Corporation*, 1994.

Paolo Ferragina and Giovanni Manzini. Opportunistic data structures with applications. In *Proceedings 41st Annual Symposium on Foundations of Computer Science*, pages 390–398, 2000. URL https://doi.org/10.1109/SFCS.2000.892127.

Ben Langmead and Steven Salzberg. Fast gapped-read alignment with Bowtie 2. *Nature Methods*, 9:357–359, 2012. URL https://doi.org/10.1038/nmeth.1923.

Brent Pedersen, Kenneth Eyring, Subhajyoti De, Ivana Yang, and David Schwartz. Fast and accurate alignment of long bisulfite-seq reads. arXiv:1401.1129 [q-bio.GN], 2014. URL https://doi.org/10.48550/arXiv.1401.1129.

Heng Li and Richard Durbin. Fast and accurate short read alignment with Burrows–Wheeler transform. *Bioinformatics*, 25(14):1754–1760, 05 2009. ISSN 1367-4803. URL https://doi.org/10.1093/bioinformatics/btp324.

Heng Li. Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM. arXiv:1303.3997 [q-bio.GN], 2013. URL https://doi.org/10.48550/arXiv.1303.3997.

Marius Wöste, Elsa Leitão, Sandra Laurentino, et al. wg-blimp: an end-to-end analysis pipeline for whole genome bisulfite sequencing data. *BMC Bioinformatics*, 21(169), 2020. URL https://doi.org/10.1186/s12859-020-3470-5.

Johannes Köster and Sven Rahmann. Snakemake—a scalable bioinformatics workflow engine. *Bioinformatics*, 28(19):2520–2522, 08 2012. ISSN 1367-4803. URL https://doi.org/10.1093/bioinformatics/bts480.

Wei Shen, Shuai Le, Yan Li, and Fuquan Hu. SeqKit: A Cross-Platform and Ultrafast Toolkit for FASTA/Q File Manipulation. *PLoS ONE*, 11 10(e0163962), 2016. URL https://doi.org/10.1371/journal.pone.0163962.