2019 27th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)

MethylFASTQ: a tool simulating bisulfite sequencing data

Giulia Piaggeschi*
Dept. of Computer Science
University of Turin
Turin, Italy
giulia.piaggeschi@unito.it

Nicola Licheri*
Dept. of Computer Science
University of Turin
Turin, Italy
licheri@di.unito.it

Greta Romano
Dept. of Computer Science
University of Turin
Turin, Italy
grromano@unito.it

Simone Pernice
Dept. of Computer Science
University of Turin
Turin, Italy
pernice@di.unito.it

Laura Follia
Dept. of Computer Science
University of Turin
Turin, Italy
laura.follia@unito.it

Giulio Ferrero
Dept. of Computer Science
University of Turin
Turin, Italy
giulio.ferrero@unito.it

***Abstract*—DNA methylation is a DNA modification playing an important role in several diseases, including cancer. The gold-standard technique for measuring DNA methylation is Bisulfite Sequencing (BS). The treatment with bisulfite alters the sequence of DNA making the analysis of BS data computationally difficult. There are many tools for analysing BS data but the choice of which to use is difficult due to the extensive biological and technical variability of the data. Synthetic and real datasets can be exploited to evaluate the tool performance and to obtain an accurate data analysis. Today, Sherman is the only available tool to generate BS synthetic datasets. However, this tool does not report any information about the methylated cytosines.**
For this purpose, in this paper we present MethylFASTQ, an easy-to-use bioinformatics tool that generates synthetic bisulfite datasets in FASTQ format. MethylFASTQ works in parallel manner using producer-consumer approach. It returns:
i) a complete dataset in FASTQ format simulating the results of a BS experiment
ii) a report file storing the information about the methylation level of the dataset (i.e. methylated cytosines).
First, we test MethylFASTQ performances with an increasing number of concurrent processes and we report the comparison of MethylFASTQ with respect to Sherman tool. Then, we also describe an application of synthetic datasets generated with our tool and we use them as input for two bisulfite mapping and methylation calling tools.
Finally, we propose MethylFASTQ as a tool to generate synthetic bisulfite sequencing data.

***Index Terms*—DNA methylation, Next Generation Sequencing (NGS), synthetic dataset, parallel computing**

I. Introduction

DNA methylation (DNAm) is the addiction of a methyl group to a DNA molecule. The DNA sequence is composed by four bases: adenine (A), thymine (T), cytosine (C) and guanine (G). The most common form of DNA methylation is the methylation of cytosine which form the 5-methylcytosine (5mC) and it affects a high number of cytosines present in the genome [1]. Methylation changes the activity of DNA without changing its base sequence.

The changes in patterns and levels of DNA methylation are associated with several diseases as cancer and genetic disorders [2]. The gold-standard technique used to study DNAm is the Whole Genome Bisulfite Sequencing (WGBS) that allows to measure methylation in the whole human genome. Conversely, targeted bisulfite sequencing (targeted-BS) allows to sequence the specific genomic regions. Both approaches belong to Next Generation Sequencing (NGS) techniques, a set of advanced technologies that allow the identification of a DNA sequence. The bisulfite treatment converts unmethylated Cs into Ts, while the other bases remain unaffected. Bisulfite conversion alters about 90% of cytosines present in the genome. At this point, distinguishing between Cs converted into Ts and a Ts originally present in the DNA molecule is computationally demanding [1]. On top of that, it is difficult to distinguish a converted C from: i) a stochastic sequencing error occurring during all the sequencing steps; ii) a Single Nucleotide Polimorfisms (SNPs). SNPs are base mutations of the genome that differ among individuals. The presence of SNPs in the samples increases the level of variability of the above data.

Since BS experiments are time and money consuming, the use of synthetic sequencing data (i.e. the creation of a dataset that simulates different biological and technical situations of a BS experiment) has become increasingly popular for assessing and validating bioinformatics tools. Simulations can also be used to evaluate software performances, for debugging purposes and to develop new computational tools [3].

II. Related works

The bioinformatics tools can be benchmarked using real and/or synthetic sequencing data. However, tools validation with real data is essential. Unfortunately, this is a difficult task because the true positive values are unknown and they

* These authors contributed equally

2377-5750/19/$31.00 ©2019 IEEE
DOI 10.1109/PDP.2019.00056

are masked by the extensive biological noise and by the variability of the data. These limitations complicate the use of real data for assessing the accuracy of tools and other performance measures [3]. Synthetic data generator tools allow the production of data with predefined parameters by defining the true positive values.

Furthermore, synthetic datasets allow the generation of a high volume of data in an inexpensive and fast way compared to costs and time needed to create real datasets in laboratory.

Synthetic data generators create FASTQ files starting from a given reference genome. FASTQ file is the *de facto* standard format to store biological data that are sequenced by NGS techniques. FASTQ format describes each read (i.e. substring of DNA) through three fields: the **sequence id** that specifies the unique identifier of the read; the **base sequence** that is the ordered sequence of bases; and the **quality score** that is a measure of quality associated to each base of the sequence.

Synthetic data generators allow to specify a variety of parameters, such as the NGS technique, the read length, the sequencing mode, the coverage and quantity of sequencing errors. The coverage parameter represents the number of times that a single base is sequenced or the number of reads aligned over a single base.

In literature there are several tools that simulate NGS data in FASTQ format, such as ART [4] and CuReSim [5]. However, tools for BS data are still lacking. At the best of our knowledge, *Sherman* is the only one tool that allows to simulate bisulfite sequencing [6]. *Sherman* is a Perl script that generates bisulfite sequencing data in FASTQ format.

Sherman allows the creation of single- and paired-end reads. The number of reads, their length and read quality can be set as tool parameters. SNPs and sequencing errors can also be set and specified. Bisulfite conversion can be regulated with two parameters, which provide the conversion rate in specific DNA contexts (i.e CG and non-CG contexts).

III. METHYLFASTQ

A. Tool overview

MethylFASTQ is a tool written in Python that generate synthetic bisulfite sequencing data in FASTQ format. It is highly customizable because MethylFASTQ is organism-independent and experiment-independent. MethylFASTQ is designed to simulates the sequencing process, following the bisulfite sequencing experiment work-flow (Figure 1).

Given a reference genome sequence as input, the user can create single-end or paired-end reads of directional and non-directional NGS libraries. The single-end mode consists in the production of one read in one direction (i.e. Forward read) for each DNA fragment. Otherwise, the paired-end mode consists in the production of two reads in two directions (i.e. Forward and Reverse reads) for each DNA fragment.

In the non-directional protocol, all four possible bisulfite DNA fragments are sequenced at the same frequency. In the directional protocol, the sequencing reads will correspond to a bisulfite converted version of either the original forward or reverse DNA fragments.

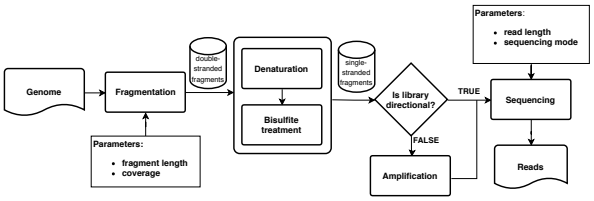

Fig. 1. **Bisulfite Sequencing workflow**. The genome of interest is fragmented in a number of double-stranded pieces of known length. Fragment strands are separated through denaturation and then, single-stranded fragments are bisulfite-treated. Amplification produces reverse complement of treated fragments, which are sequenced in the non-directional protocol. Sequencing step processes bisulfite fragments and produces a set of reads of known length.

MethylFASTQ also allows to simulate both WGBS experiment and targeted-BS data. Two files are returned: a FASTQ file(s) and a methylation call file. In case of single-end sequencing, a single FASTQ file is produced. Differently, in case of paired-end sequencing two FASTQ files are produced which contain respectively the forward and reverse reads. The methylation call file contains the information about the sequenced cytosines.

Two experimental modes are implemented: 1) in the WGBS mode the user can optionally provide a list containing the chromosome names that have to be sequenced. If no list is provided the entire reference genome will be sequenced; 2) in targeted-BS mode the user must provide a tabulated file containing the genome regions to be sequenced. This file will contain the chromosome number and the indexes of first and last base for each region that will be sequenced. Moreover, the user may define the fragment size (i.e the reads length) and the depth of coverage. Methylation can be set through three context-based probabilities: CG, CHG and CHH (where H= A,T or C). The user can also specify probabilities about SNPs and sequencing errors. All the reads which cover a specific base will report the mutated base with a quality is not discernible from a non-mutated base.

Each read in the FASTQ file has an unique record "id" which provides information about its true mapping position in the reference genome. Specifically, the record "id" of a generic read has the form **chr:pos:strand**, where:

- **chr** is the chromosome from which the fragment has been extracted;
- **pos** is the position of the first base in the chromosome;
- **strand** identifies the DNA strand. It can be either forward (+) or reverse (-);

Regarding the methylation call file, it is a file which presents a line for each covered cytosine. Each line has the form **chr pos strand ctx nmeth ntot beta**, where:

- **chr** is the chromosome in which the cytosine is located;
- **pos** is the index of the cytosine in the chromosome (starting from 0);
- **strand** is the strand, it can be either forward (+) or reverse (-);
- **ctx** is the cytosine context, it can be either CG, CHG or CHH;

- **nmeth** represents how many times the cytosine appears as methylated;
- **ntot** represents how many times the base was sequenced;
- **beta** is the beta value of cytosines, defined as the ratio $nmeth/ntot$.

B. Software architecture

MethylFASTQ is modularized in three different modules.

1) `methylfastq` module contains the list of command line arguments and the main class `MethylFASTQ`. This class checks the input parameters and reads the input reference genome file, starting sequencing either in WGBS mode or targeted-BS mode.
2) `sequencing` module implements the sequencing procedures by means of two classes. The first class, called `ChromosomeSequencer`, splits an entire chromosome record in subsequences. These are independently sequenced by the second class, called `FragmentSequencer`.
3) `dna` module contains auxiliary classes that implement different types of DNA sequences, such as double- and single-stranded fragment or single- and paired-end reads.

MethylFASTQ architecture follows the well-known **producer-consumer** software design pattern. The *producer*'s job (Figure 2) is to generate the data and to send it to the consumer. Conversely, the *consumer* (Figure 3) has to consume the received data one at time. Parallelization is process-based and utilizes the built-in module `multiprocessing`, which supports spawning processes and assigning them a job through a function. Inter-process communication is performed using a FIFO queue implemented in `multiprocessing` module, which is process-safe and thread-safe. A process attempting to get an element from an empty queue is blocked until an element is available. In a similar way, a process attempting to put an element in a full queue is blocked until a free slot is available.

The parent process acts as the consumer, whereas the producers are represented by the child processes.
MethylFASTQ works with a chromosome sequence at a time. Chromosome substrings separated by unspecified bases, represented by 'N' characters, are located and extracted. Extracted substrings are split in order to equally distribute the workload among a number of parallel processes.

The load balancing step starts by calculating the total size of the extracted substrings and their average length ($\bar{m}$) that should be assigned to each process. Sequences length $\hat{m} \geq \bar{m}$, longer than the average value, are splitted into M substrings of length $\bar{m}$ and one of length r, where $M,\ r$ are chosen such that $\hat{m} = \bar{m} \cdot M + r$ with $0 \leq r < M$.

The resulting substrings are sorted with respect to their length in descending order, so that shorter substrings will be processed after the longer ones.
Finally, the user can define a set of processes (workers) that will elaborate the substrings. Sequences with their offsets are distributed among the workers and sequenced in a parallel manner.

Data generated by the workers can be of three types:

1) a list of single-end reads in FASTQ format;
2) a list of paired-end reads in FASTQ format, where the generic paired-end read is a pair;
3) a list storing the methylation information about covered cytosines of the sequenced substring;

so that each kind of data can be stored in a different file.

Workers instantiate a `FragmentSequencer` object using as input parameters the chromosome substring and its initial and final offsets. Random SNPs are set on the string, using the SNP rate parameter given by the user. Then, cytosines on both strands of the sequence are indexed. Cytosines information are stored in a hash table, where the cytosine position into the fragment acts as a key and a `Cytosine` object is the corresponding value. This object contains the strand and context information, as well as two values that take into account how many times that base is covered by a read, and how many times it appears methylated.

Numerous overlapping fragments are extracted from the sequence, so that each base is covered (on average) by a number of reads equal to the chosen depth of coverage. A methylation is generated w.r.t. a probability based on the context (CG, CHG, CHH). Single- or paired-end reads, depending on the chosen sequencing mode, are then extracted from bisulfite strands and stored into a buffer. If the non-directional library has been chosen, reads are also extracted from reverse complement of the bisulfite fragment strands. Whenever the number of reads in the buffer is greater than a certain threshold, it is flushed in the shared queue, so that the parent process can permanently store them in a file. Reads generation involves sequencing error set up and the creation of the relative FASTQ record. Setting up the sequencing errors changes each base with a probability given as input. Quality score associated to changed bases is drastically lowered.
FASTA file scanning and FASTQ record creation are accomplished using BioPython package [7].

IV. Results

In this section are described the results from: (1) the application of MethylFASTQ to generate different synthetic datasets with associated execution times; (2) the comparison between MethylFASTQ and *Sherman* tools performances; (3) the application of MethylFASTQ synthetic datasets in the BS analysis pipeline performed using two BS data mapping and methylation caller tools (BSMAP [8] and Bismark [9]). The experiments were performed on a 48-core AMD Opteron 6176 CPUs at 2.3 GHz with 503 GB of RAM.

A. MethylFASTQ performances

The measure of the execution time is an indicative quantification of software performance. Indeed, the time needed to complete a task is dependent on the machine workload.

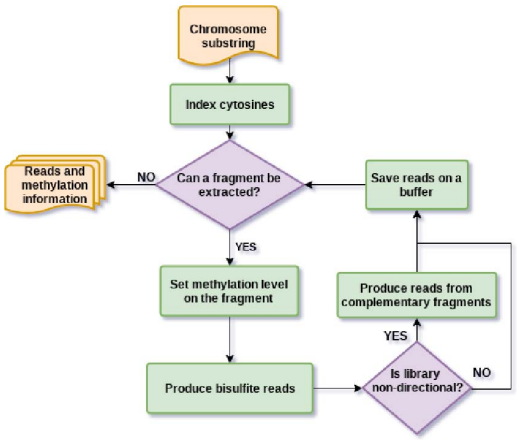

Fig. 2. **Producer process**. Cytosines of the chromosome substring are indexed. Several overlapping substrings are extracted from the chromosome substrings. For each of them, methylation is set and relatives information are stored in the index. Then, the bisulfite fragment is produced and reads are extracted from it. Reads are stored in a local buffer which is periodically flushed in the queue. When fragments extraction terminates, the consumer pushes in the queue the cytosines information and its execution ends.

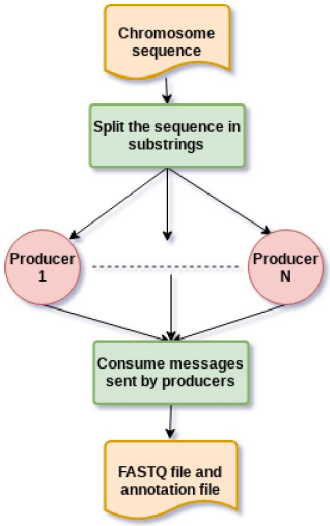

Fig. 3. **Consumer process**. The chromosome sequence is splitted in non-overlapping substrings, which are further divided by the load balancing algorithm. Obtained substrings are assigned to N producer processes. Then, the consumer waits for items to be available in the queue and elaborate them. When all substrings have been sequenced, the consumer terminates.

As reported in Table I the average execution time for the generation of each dataset increases in proportion with the features complexity. Indeed, the lower execution time was obtained for creating the dataset with single-end reads of directional library while the generation of paired-end reads of non-directional library was the most expensive execution. As reported in Figure 4 the MethylFASTQ execution time rapidly drops as the number of parallel processes increases. The execution time using one process was longer than ten hours, while with two processes the execution time was halved, and finally dropped to minutes with seven and eight processes.

Sequencing	Library	Generation time (min)
single-end	directional	15
single-end	non-directional	24
paired-end	directional	25
paired-end	non-directional	44

TABLE I
AVERAGE TIME COMPUTED CONSIDERING 10 RUNS USED TO CREATE THE DATASETS USING EIGHT PARALLEL PROCESSES. ALL THE DATASETS ARE EXTRACTED FROM CHROMOSOME 21 OF HG19 REFERENCE AND HAS 10X COVERAGE. FOR EACH EXPERIMENT, 10 METHYLFASTQ EXECUTIONS HAVE BEEN PERFORMED AND THE AVERAGE TIME HAS BEEN CALCULATED. TIMES ARE EXPRESSED IN MINUTES.

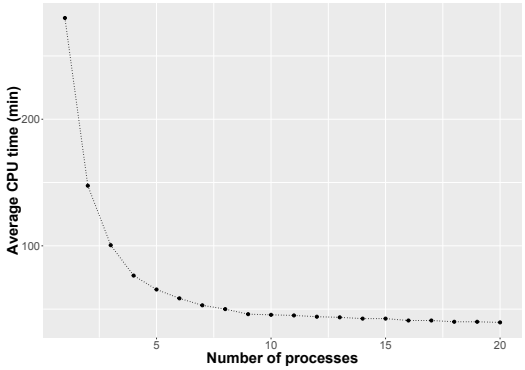

Fig. 4. **MethylFASTQ execution times performances**. Average time of 10 runs to create a dataset as the number of parallel processes increases. The dataset is extracted from human chromosome 21. It is a non-directional library with paired-end reads with 10x coverage.

B. Comparison between MethylFASTQ and Sherman tools

We compared the performance of MethylFASTQ and the already published *Sherman* tool [6] (Figure 5). Both tools generate bisulfite synthetic data in high customizable way and they allow the setting of the reads length, the single-end/paired-end mode and the directionality of the libraries. In addition, they allow the setting of the bisulfite conversion rate for all the cytosines and the simulation of different reads quality scores as well as the number of random SNPs in each read. The final output of both these tools is a FASTQ file, however, Sherman does not produce a report file related to methylation calling for each sequenced cytosine. *Sherman* also does not allow the simulation of a targeted-BS experiment but only a WGBS, because it is not possible to select a set of specific fragments from the reference genome.
The results of the tools comparison show that when both tools run with one process *Sherman* performs better in terms of execution time than MethylFASTQ (Figure 5). This is probably due to the double step of MethylFASTQ that is:
(i) apply the methylation function on genome substrings and save them
(ii) produce a report file storing the information of data methylation profile.
Since *Sherman* is not a parallel tool, the below comparison of

execution times will show the performances of MethylFASTQ using up to eight processes, while *Sherman* runs in sequential mode. The results are different when MethylFASTQ runs with an increasing number of processes. Indeed, the run of MethylFASTQ with two processes obtains comparable execution time with respect to *Sherman*. Instead, with a further increase of the processes number, MethylFASTQ performs better than *Sherman*, due to the parallelization.

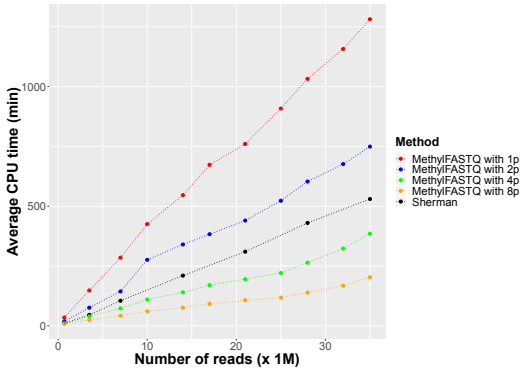

Fig. 5. **Comparison between *Sherman* and MethylFASTQ tools.** Average times to produce datasets of seven different sizes by *Sherman* and MethylFASTQ. MethylFASTQ has been run with 1, 2, 4 and 8 producer processes. Datasets were extracted from human chromosome 21 of human genome hg19. They are non-directional libraries with paired-end reads.

C. MethylFASTQ helps on the comparison of bisulfite aligners and methylation callers

The synthetic datasets generated with MethylFASTQ were used as input for a comparative analysis between BSMAP [8] and Bismark [9] performances on the alignment and the methylation calling tasks. These tools follow two different approaches for BS reads mapping: BSMAP applies an approach based on the hashing technique; it masks cytosines in the reference genome to allow bisulfite mismatches. Conversely, Bismark converts both reads and reference in 3-letter sequences and then it applies an algorithm based on the Burrows-Wheeler transform [10]. Methylation calling is performed by methylation extractors included in BSMAP and Bismark packages. All the tools have been tested using their default settings.

The alignment percentage and the recall on identified CG sites were used as performance measurements. The **alignment percentage** considers only the uniquely mapped reads (i.e those reads that are mapped in only one position with a minimum number of mismatches). In case of paired-end reads the reads are aligned if both the extremities are properly mapped. The **recall** is the fraction of true positive values correctly identified as methylated CG sites. It is defined as: TP/Pos, where, TP is the number of CG sites identified by the tool and Pos is the total number of CG sites.

Ten synthetic datasets with different combinations of parameters have been generated to evaluate the tools performances as the library settings and the reads quality level change (Table II).

	Sample	Aligned reads		Recall	
ID	num. reads	BSMAP	Bismark	BSMAP	Bismark
SD1	7.024.152	98.45%	98.68%	98.19%	99.13%
SD2	7.023.824	98.53%	98.37%	93.88%	99.10%
SD3	7.018.280	98.58%	94.95%	89.41%	99.13%
SD4	7.019.916	98.04%	41.37%	95.36%	93.89%
SD5	7.021.892	98.46%	98.65%	96.46%	97.40%
SD6	7.016.484	98.50%	98.55%	94.51%	97.34%
SD7	7.017.776	98.47%	98.58%	94.80%	95.74%
SD8	7.017.556	98.55%	95.55%	89.18%	88.40%
SD9	7.021.028	93.52%	15.46%	74.32%	49.82%
SD10	7.022.140	94.86%	19.77%	63.11%	37.91%
min	7.016.484	93.51%	15.46%	63.11%	37.91%
max	7.024.152	98.58%	98.68%	98.19%	99.13%
avg	7.020.305	97.6%	76%	88.92	85.79%

TABLE III
ALIGNMENT AND METHYLATION EXTRACTION PERFORMANCES ON THE SYNTHETIC DATASETS. MAPPING AND METHYLATION CALLING RESULTS ON SYNTHETIC DATASETS OF BSMAP AND BISMARK TOOLS.

The comparison between alignment performances using these synthetic datasets show that BSMAP is stable as the sequencing error rate or the presence of SNPs increases (Table III). The alignment percentages have little variability, even for low quality datasets. Conversely, Bismark alignment performances vary dramatically with the increase of sequencing errors/SNPs rate. However, the alignment performances have not a great impact on the methylation extraction. Indeed, using low quality datasets with associated low alignment percentages, the methylation extraction works properly. An example is the synthetic dataset 9 (SD9) for which Bismark aligns only 15% reads obtaining a recall of 50% (Table III).

ID	num. reads	SNP rate	Error rate	num. CG sites
SD1	7.024.152	0.1%	0.1%	766.422
SD2	7.023.824	0.1%	1.0%	766.748
SD3	7.018.280	0.1%	2.0%	766.398
SD4	7.019.916	0.1%	5.0%	766.698
SD5	7.021.892	0.3%	0.1%	777.718
SD6	7.016.484	0.3%	0.5%	778.154
SD7	7.017.776	0.5%	0.1%	789.096
SD8	7.017.556	1.0%	1.0%	817.514
SD9	7.021.028	2.0%	5.0%	873.480
SD10	7.022.140	5.0%	2.0%	1.038.142

TABLE II
CONSTRUCTION PARAMETERS OF THE USED SYNTHETIC DATASETS. ALL THE DATASETS ARE EXTRACTED FROM CHROMOSOME 21 OF HG19 REFERENCE. THEY ARE NON-DIRECTIONAL DATASETS WITH PAIRED-END READS OF LENGTH 150 BASES USING A 10X COVERAGE. DATASETS WERE GENERATED FROM HUMAN CHROMOSOME 21 OF HUMAN GENOME HG19.

V. CONCLUSION

In this paper we present MethylFASTQ a new parallel tool to generate bisulfite synthetic datasets. MethylFASTQ allows us to generate both reads and a report file of methylation call, which contains information about methylated cytosines. We showed that our tool helps to find the weaknesses of two mapping and bisulfite caller tools, Bismark and BSMAP. In the future, we will implement MethylFASTQ in C/C++ language

in order to switch from multiprocessing to multithreading, enhancing software performances.

AVAILABILITY AND IMPLEMENTATION

MethylFASTQ is released under the GNU GPLv3 license. It is freely available at https://github.com/qBioTurin/MethylFASTQ.

REFERENCES

[1] Katarzyna Wreczycka, Alexander Gosdschan, Dilmurat Yusuf, Björn Grüning, Yassen Assenov, and Altuna Akalin. Strategies for analyzing bisulfite sequencing data. *Journal of biotechnology*, 261:105–115, 2017.

[2] Christoph Bock. Analysing and interpreting dna methylation data. *Nature Reviews Genetics*, 13(10):705, 2012.

[3] Merly Escalona, Sara Rocha, and David Posada. A comparison of tools for the simulation of genomic next-generation sequencing data. *Nature Reviews Genetics*, 17(8):459, 2016.

[4] Weichun Huang, Leping Li, Jason R Myers, and Gabor T Marth. Art: a next-generation sequencing read simulator. *Bioinformatics*, 28(4):593–594, 2011.

[5] Ségolène Caboche, Christophe Audebert, Yves Lemoine, and David Hot. Comparison of mapping algorithms used in high-throughput sequencing: application to ion torrent data. *BMC genomics*, 15(1):264, 2014.

[6] F. Krueger. Sherman - bisulfite-treated Read FastQ Simulator. https://www.bioinformatics.babraham.ac.uk/projects/sherman/. Accessed: 2018-09-20.

[7] P. J. A. Cock et al. Biopython: freely available Python tools for computational molecular biology and bioinformatics. *Bioinformatics*, 25(11):1422–1423, mar 2009.

[8] Yuanxin Xi and Wei Li. Bsmap: whole genome bisulfite sequence mapping program. *BMC bioinformatics*, 10(1):232, 2009.

[9] Felix Krueger and Simon R Andrews. Bismark: a flexible aligner and methylation caller for bisulfite-seq applications. *bioinformatics*, 27(11):1571–1572, 2011.

[10] Micheal Burrows and David Wheeler. A Block-Sorting Lossless Data Compression Algorithm. Technical report, DIGITAL SRC RESEARCH REPORT, 1994.