HMST-Seq-Analyzer: A New Package for Differential Methylation Analysis of Whole-Genome Methylation Data

Sindre Grønmyr

Thesis submitted for the degree of
Master in Informatics: Technical and Scientific Applications
60 credits

Department of Informatics
Faculty of mathematics and natural sciences

UNIVERSITY OF OSLO

Autumn 2019

HMST-Seq-Analyzer: A New Package for Differential Methylation Analysis of Whole-Genome Methylation Data

Sindre Grønmyr

HMST-Seq-Analyzer: A New Package for Differential Methylation Analysis of Whole-Genome Methylation Data

http://www.duo.uio.no/

Printed: Reprosentralen, University of Oslo

Abstract

DNA methylation is essential for normal development and many different biological processes in genomic DNA. It is also associated with the development of diseases when there are abnormal methylation patterns. For example, hypomethylation in certain cells could lead to chromosomal instability and oncogene activation. Hypermethylation is also commonly known to silence certain tumor suppressor genes. Thus, the study of DNA methylation is an active field in cancer epigenomics research.

This thesis presents a new high-level analysis pipeline, called *HMST-Seq-Analyzer*, which can do differential methylation analysis on HMST-Seq datasets. The HMST-Seq technique detects single-base resolution 5-methylcytosine (5mC) and 5-hydroxymethylcytosine (5hmC). The new Python package can simultaneously analyze 5mC and 5hmC data from single HMST-Seq experiments. The pipeline predicts hypo/hyper differential methylated/hydroxymethylated regions in predefined genomic regions, such as TSS, gene body, TES, intergenic, 5'-UTR, and enhancer. As a part of the results, the pipeline creates figures to illustrate useful information from the data analysis done in various parts of the pipeline. HMST-Seq-Analyzer is able to analyze either mouse or human data. Especially, it can be used to study DNA methylation data generated by other platforms such as whole-genome bisulfite sequencing (WGBS). The new pipeline is tested on both public human HMST-Seq data from the ENCODE project and an in-house mouse sequencing.

In the end, we compare the results of HMST-Seq-Analyzer with that of another popular DNA methylation analysis tool - methylKit, where the majority of predictions are overlapping between the two methods. It proves that HMST-Seq-Analyzer is a reliable tool for differential analysis of DNA methylation sequencing.

Acknowledgements

I would first like to thank my supervisor, Junbai Wang, for involving me in an interesting project, for always being available answering my questions, and for providing guidance throughout the entire process. I would also like to thank Torbjørn Rognes for reading through my thesis and giving helpful advice. Additionally, apologies go out to my girlfriend, friends, and family for my absence during parts of 2019.

Contents

List of Figures

List of Tables

Part I

Introduction

Chapter 1

Motivation

Nowadays, high-throughput sequencing technologies are widely utilized in the biomedical research field. It is easy to generate tons of genome sequencing datasets from various experiments such as ChIP-seq, RNA-seq, whole-genome/exome sequencing, and DNA methylation sequencing. However, it is very challenging to analyze large datasets from various high-throughput genome experiments.

There are already multiple tools available for analyzing DNA methylation data. Many of them are made for data generated from bisulfite sequencing (BS-Seq), but since BS-Seq does not differentiate 5mC and 5hmC, we cannot use these tools for the current project. There are also tools developed for analyzing both 5mC and 5hmC data, but it is difficult to use these existing tools to analyze both datasets simultaneously. Thus, in this project, we intend to build a new computational analysis pipeline by considering a proper statistical method for analyzing 5mC and 5hmC datasets obtained from whole-genome methylation sequencing experiments. The pipeline will be made as a command-line tool, written in the high-level programming language Python. The analysis will include finding differentially methylated and hydroxymethylated regions, which is an essential step in analyzing DNA methylation samples.

Chapter 2

Background

2.1 DNA

2.1.1 Nucleotides

Every living organism, from bacteria to plants to humans, possesses a genome that contains biological information of the organism. Most genomes, such as human, are made of deoxyribonucleic acid, or DNA. DNA is a polymer molecule consisting of multiple subunits. These subunits, called nucleotides, are chemically distinct and are linked together to form long chains. There are four types of nucleotides: adenine (A), cytosine (C), guanine (G) and thymine (T). Each nucleotide is built from three components: a sugar, a phosphate group, and a nitrogenous base, which decides the A, C, G or T for the nucleotide [1]. This is illustrated in Figure 2.1.

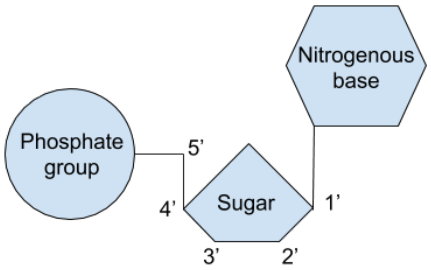

Figure 2.1: The three components of a nucleotide: a phosphate group, a sugar and a nitrogenous base. The sugar consists of five carbon atoms, 1′ to 5′.

2.1.2 Double Helix

Individual nucleotides are bonded together to form a chain of single nucleotides, a polynucleotide, where nucleotides are linked together in a way that the two ends are chemically distinct. The polynucleotide has a chemical direction that is expressed either as 3′ -> 5′ or 5′ -> 3′. That is because the sugar of the nucleotides consists of five carbon atoms, named 1′ to 5′ respectively, which can be seen in Figure 2.1. It is always linked

between the 3′ carbon of the first nucleotide to the 5′ carbon of the second when a nucleotide is bonded to the other one. Two polynucleotide strands are linked together to form a double helix, by pairing the base A with T or the base C with G, as shown in Figure 2.2. The DNA strands are then said to run anti-parallel because the 5′ end of one strand is parallel with the 3′ end of the other strand [1].

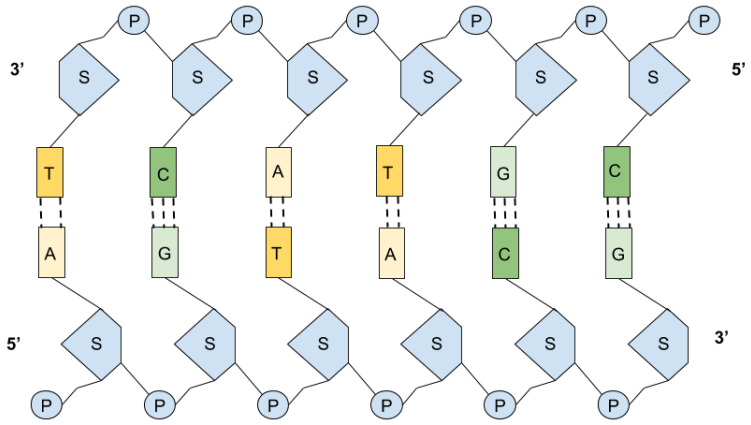

Figure 2.2: The double helix structure, with sugar (S), phosphate (P), nitrogenous bases (A, C, G, and T), the hydrogen bonds between the nitrogenous bases (dashed lines), and how they are connected.

2.1.3 Genes

The most important feature of a DNA molecule is the nucleotide sequence and especially the genes, which encode functions. Most human genes are discontinuous, which means the coding regions and exons are split between non-coding regions called introns. It is the information that is encoded in the genes, that is used to control gene regulation [1].

2.1.4 Gene Transcription

Gene expression, which is the process of converting the information encoded in the genes into a functional product, such as proteins, has three essential stages: transcription, splicing, and translation. In this project, we will mostly focus on the transcription part of gene regulation. The regulation of gene expression represents mechanisms that repress or induce the expression of a gene. For example, in transcription, the two strands of the DNA double helix is separated by an enzyme (RNA polymerase) at a promoter region, where the RNA polymerase reads one strand of the DNA and copies the other strand of nucleotides into a complementary pre-mRNA strand, and thymine is replaced with uracil (U).

2.1.5 Genomic Regions

In a gene, there are different genomic regions. In this project, we will focus on transcription start site (TSS), gene body, transcription end site

(TES), 5′-untranslated distance region (5′-UTR), and intergenic regions. In addition to these five genomic regions, we will also study enhancers. A short description of the different genomic regions is shown here.

Transcription Start Site

The transcription start site, or TSS, is the location where transcription starts by RNA polymerase transcribing the first DNA nucleotide into RNA. In this project, the TSS is an upstream region of a TSS, or the beginning of a gene.

Gene Body

The gene body is the transcriptional region that is transcribed into mRNA, i.e., between TSS and TES. This region includes both introns that will be removed from the mRNA, and exons, which will be translated into protein.

Transcription End Site

The transcription end site, or TES, is the location that marks the end of transcription. In this project, the TES is a downstream region to the end of a gene.

5′-Untranslated Region

The pre-mRNA, in addition to a gene, contains sequences from the regions preceding the first exon and following the last exon. These regions are called the 5′-untranslated region (5′-UTR) and 3′-untranslated region (3′-UTR) [1], respectively. A part of the 5′-UTR named Upstream Open Reading Frame (uORF), even though called untranslated, can sometimes be translated into a product. uORFs may also affect gene expression by altering mRNA stability [2].

Intergenic Region

The intergenic region is another type of non-coding region but is located in between the genes. Earlier, scientists thought all non-coding DNA was "junk" that did not have any purpose, but it is now known that these regions also contain important parts that have essential functions. One of the functions of non-coding regions, including intergenic regions as well as introns, is the control of gene activity, such as promoters, enhancers, and silencers. Of these three regulatory elements, only enhancers will be used in this thesis.

Enhancer Region

Enhancers are short genetic regions that, when bound by transcription factors, increases the likelihood that transcription of an associated gene will occur. They are functional regulatory elements, which are important

in gene regulation during mammalian development [3]. Although mammalian genomes potentially contain millions of enhancers, only some of them are active in a given cell type or tissue. Furthermore, their location relative to the target gene can be very variable, e.g., enhancers can be both upstream, downstream, or within introns of target genes. They are also located at distant regions respective to the target genes. The general sequence of enhancers is, contrary to the sequences of protein-coding genes, poorly understood. Because of this, the target genes of enhancers are difficult to be identified from DNA sequences [4]. Previously, enhancers were assigned to the nearest transcription start sites. However, Y. Shen et al. [3] developed an algorithm for detecting local clusters of co-regulated promoters and enhancers, defined as enhancer-promoter units (EPUs). In this thesis, our predicted enhancers are based on the EPUs method.

2.2 DNA Methylation

Epigenetics is a term that describes processes modifying gene activity without changing the actual DNA sequence. The most well studied epigenetic process is DNA methylation [5], where the cytosine bases in DNA molecules are changed to 5-methylcytosine (5mC) by adding a methyl-group (CH_3) [1]. The addition of CH_3 often occurs at CG sites, which is where a cytosine is directly followed by a guanine, reading from 5′ to 3′. DNA methylation mostly contributes to repress or silence gene regulation, which means a possible gene expression will not occur if there is a high level of methylated cytosines in the gene promoter region. It also affects phenotypic variations [6], which are the observable and measurable physical traits of an organism [7], such as eye color, height, or the sound of one's voice.

For normal development and many different biological processes, the 5mC in genomic DNA is essential [8]. Also, DNA methylation is stable in most of the CG islands (CGIs), which do not change methylation state during normal development. CGIs are regions containing a high frequency of CG sites and are clusters of typically a couple of hundred and more base-pairs in length [9]. However, DNA methylation is associated with the development of diseases when there are abnormal methylation patterns: for example, hypomethylation in cells known to be relevant for cancer could lead to chromosomal instability and oncogene activation [10]. The oncogene activation is "turning on" a gene that has the potential to cause cancer. Hypermethylation is also commonly known to silence certain tumor suppressor genes [6]. Thus, DNA methylation study is an active field in cancer epigenomics research.

One critical step in analyzing DNA methylation data is differential methylation analysis, e.g., by discovering differentially methylated regions (DMRs) between multiple samples, such as a tumor versus a normal sample [11]. DMRs are regions that have significantly different methylation patterns between samples. Many statistical methods can be used to find this [6, 12].

Usually, 5mC can be oxidized to 5-hydroxymethylcytosine (5hmC) by ten-11 translocation (TET) enzyme family proteins. 5hmC is found in many tissues and cell types. Its functional role is to reduce local 5mC levels (e.g., demethylation) and activate the state of a gene. Therefore, 5hmC also plays a vital role in gene regulation. Though it can be further oxidized to 5-formylcytosine (5fC) and 5-carboxylcytosine (5caC) by TET proteins [8], the main focus of the current study will be differential analysis of 5mC and 5hmC data.

2.2.1 Detection of DNA Methylation by Using Hydroxymethylation and Methylation Sensitive Tag Sequencing - HMST-Seq

F. Wang et al. [11] presented an overview of the different technologies for detecting DNA methylation, but all of these were made before the discovery [13] of the hydroxylated form of 5mC, 5hmC. Since both 5mC and 5hmC are of considerable biological importance, they need to be distinguished unambiguously for a full appreciation [8].

Though there are existing methods for detecting both 5mC and 5hmC, they suffer from low resolution and are biased towards experiment, meaning that they are experiment-specific and may only be suited for a particular type of experiment. Methods that can map 5hmC at genome-wide level with single-base resolution, e.g., oxBS-Seq and TAB-Seq, have been made, but they require massive amounts of data to determine both 5mC and 5hmC in parallel for the human genome. In theory, one would need 180 GB sequenced data to achieve 30x sequencing depth [8]. Sequencing depth, or coverage, is the average number of reads that overlap with a region when sequencing, as can be seen in Figure 2.3. Therefore, a cost-effective and high-resolution strategy was made, called hydroxymethylation and methylation sensitive tag sequencing (HMST-Seq). It provides a method for detecting single-base resolution 5mC and 5hmC in MspI sites (5'-CCGG-3') in the human genome. In theory, it only requires 5.67 GB data to achieve 30x sequencing depth [8], which decreases the sequencing cost significantly. HMST-Seq is the method that is used in this project to generate methylation data.

```
Read 1:  CGGATTACGTGGACCATG
Read 2:     ATTACGTGGACCATGAATTGCTGACA
Read 3:               ACCATGAATTGCTGACATTCGTCA
Read 4:                   TGAATTGCTGACATTCGTCAT
Depth:   11122222222233334433333333333222222221
```

Figure 2.3: An example of the coverage/depth at each nucleotide for some overlapping reads.

The HMST-Seq method creates three different types of libraries based on the same genomic DNA, called "C"-, "C + mC"- and "C + mC +

hmC"-library, where "C" is comprised of unmodified cytosines, "C + mC" of both unmodified and methylated cytosines, while "C + mC + hmC" of all unmodified, methylated and hydroxymethylated cytosines. The DNA samples go through three slightly different processes to generate the respective libraries, where the main difference is that the samples are cut by different restriction enzymes. Restriction enzymes are enzymes that cleave DNA at a specific site, as can be seen in an example in Figure 2.4.

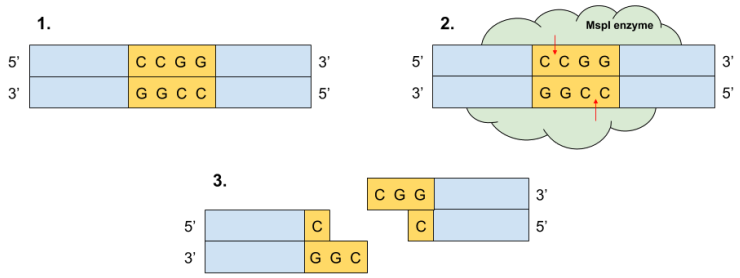

Figure 2.4: How the DNA is cut at a specific site by the MspI enzyme. We say that MspI cuts at C^CGG.

The sample that is represented in the "C + mC" library is first glucosylated, which blocks MspI from cutting 5hmC. This sample is then digested with MspI restriction enzyme. The sample represented in the "C" library is digested directly with HpaII restriction enzyme, while the "C + mC + hmC" library is made from direct MspI digestion. MspI and HpaII are isoschizomers, which means that they both recognize the same sequence, in this case, 5'-CCGG-3', the difference being that HpaII only recognizes the sequence when CG is unmethylated, while MspI can recognize all unmethylated, methylated and hydroxymethylated CGs. After the digestion, each DNA fragment goes through the same genetic engineering, which will not be discussed in detail here, before it, in the end, is sequenced. This results in short sequence tags, or reads, of 16-17 bp being obtained. These tags then go through some low-level analysis, consisting of mapping the raw sequencing tags to a virtual library consisting of a genome sequence, such as the human or mouse genome sequence. The result of mapping the sequencing tags to this virtual library is the three libraries containing tag counts, the tag counts being the number of tags, i.e., the sequencing depth in a particular genome site.

The tag counts described above can be normalized among the libraries to make the tags more robust and to give accurate comparisons of results between the different sample conditions because they might contain bias from the library construction and sequencing [8]. The normalization can, for example, be done by using an algorithm called Global rank-invariant set normalization (GRSN). This method creates a reference, called Global Rank-invariant Set (GRiS), based on the tags in the libraries and the variance at each site over the libraries. Then the library tags are normalized based on the GRiS method [14]. Since the input tags are already normalized by the GRiS method, we will implement another method called quantile

normalization [15] in the pipeline that can be used to perform similar normalization on raw tag counts. The three libraries: "C", "C + mC" and "C + mC + hmC" will from now on be referred to as HpaII, β-GT/BGT and MspI libraries respectively.

2.2.2 Detection of DNA Methylation With BS-Seq

When sequencing biological data using the HMST-Seq method, we can detect 5mC and 5hmC at CG sites with single-base resolution. However, methylation can also occur at CHG and CHH sites, where H = A, C, or T. Bisulfite sequencing (BS-Seq) can be used to detect 5mC and 5hmC with single-base resolution in all CG, CHG, and CHH context. This is done by treating the DNA sample with sodium bisulfite before sequencing. The bisulfite treatment converts unmethylated cytosines to uracils, whereas methylated forms of cytosine such as 5mC and 5hmC remain unchanged [9]. The problem with this method is that it does not distinguish between 5mC and 5hmC. However, there will be an alternative in the pipeline for doing differential methylation analysis on datasets generated by the whole-genome bisulfite sequencing (WGBS) method also.

2.3 Statistical Hypothesis Testing

When doing differential methylation analysis, there are many statistical tests that could be used. In this section, we will present a few general statistical hypothesis testing methods that can be used to find differentially methylated regions. We will also show what the multiple comparisons problem is and how to handle multiple comparisons in the current work.

2.3.1 Hypothesis Testing

Hypothesis testing refers to the process of using sample data to decide whether the null hypothesis should be rejected [16]. There are two types of statistical hypotheses: the null hypothesis, denoted H_0, and the alternative hypothesis, denoted H_1. The null hypothesis is a statement containing a zero difference, representing what is assumed to be true. It is this hypothesis that goes through the test procedure. Because the alternative hypothesis is the opposite of the null hypothesis, the alternative hypothesis must be true if the null hypothesis is false. The alternative hypothesis can be either two-tailed or one-tailed. A one-tailed hypothesis test allows us to test the statistical significance in one direction of interest, disregarding the possibility of a relationship in the other direction. However, if we use a two-tailed test, we can test the statistical significance in both directions at the same time. The strength of evidence supporting the null hypothesis can then be calculated by the p-value. A p-value is the probability of finding the observed or more extreme results under the assumption that the null hypothesis is true. The null hypothesis is then rejected if the calculated p-value is less than a chosen significance level, denoted α. Typical significance levels are 0.05 or 0.01.

Depending on whether H_0 is rejected or accepted and which of H_0 and H_1 is true, one of four possible situations will occur. They are shown in Table 2.1. Here we can see that two of the situations are correct decisions, and two are decisions that give errors. We say that a type I error occurs when we reject the null hypothesis when it is true. A type II error occurs when we do not reject the null hypothesis even though it is false [16]. Type I errors and type two errors are sometimes also referred to as false positives and false negatives, respectively.

Table 2.1: The four different situations that can occur when doing a hypothesis test. There are two types of errors that can be made: (type I error) the true null hypothesis can be incorrectly rejected and (type II error) the false null hypothesis can fail to be rejected.

Statistical Decision	True state of null hypothesis	
	H_0 True	H_0 False
Reject H_0	Type I Error	Correct
Do not Reject H_0	Correct	Type II Error

We can divide statistical hypothesis tests related to differences into two categories: parametric tests and non-parametric tests. A parametric test is a hypothesis test where there is an assumption that the sample data comes from a specific underlying distribution such as the normal distribution. Non-parametric tests, on the other hand, are used when we do not have any distributional assumptions, i.e., when we cannot assume that the data is following any specific distribution. Parametric tests are in general more powerful than non-parametric tests. For example, for rank-based non-parametric tests, we substitute ranks for the original values. We will then lose information, which makes it less powerful. However, non-parametric tests are often necessary. For example, if the distribution is skewed or unknown, or the sample size is too small (< 30) to have any distributional assumption, non-parametric tests should be used [17].

2.3.2 Wilcoxon Rank-Sum Test

The Wilcoxon rank-sum test is a non-parametric test for comparing two unpaired groups of observations [18, 19]. It is essentially identical to the Mann-Whitney U test, and the two are used interchangeably, but the Mann-Whitney U test uses a different test statistic. The goal of the Wilcoxon rank-sum test is to detect if there are any departures from the null hypothesis (H_0), where $H_0 : A = B$ and A, B are two populations containing n_A and n_B observations respectively. The Wilcoxon test is based on ranking the $n = n_A + n_B$ observations of the combined sample. We will here try to give a short overview of how the basics of this rank-based hypothesis test are. The rank is determined from the sorted, combined sample and is starting at 1, increasing based on the data point. If there are duplicate values, called ties, the ranks are adjusted to receive the median rank for the entire identically sized group. Thus, if the values of rank 1 and 2 are identical, they will both receive the rank of 1.5. The test statistic is then the

sum of the ranks for observations from one of the samples. For sample A, we will call the test statistic w_A. For sample B, we will call the test statistic w_B. The corresponding rank-sum for A when H_0 is true is called W_A, and the distribution of this can be found in a table [18, 19]. To find out if there is any evidence pointing against H_0, we test H_0 versus H_1, which can be either one of $A > B$ or $A < B$. By finding the p-value for this, calculated by: $P(W_A \geq w_A)$ for $H_1 : A > B$ or $P(W_A \leq w_A)$ for $H_1 : A < B$, we can see if there is any evidence against H_0 [18, 19]. If we have no strong prior reason for expecting a shift in one particular direction, we can use the two-sided alternative. This can be done by doubling the probability of falling into the tail of the distribution closest to w_A, such that if w_A is in the lower tail then p-value=$2P(W_A \leq w_A)$, and if w_A is in the upper tail then p-value=$2P(W_A \geq w_A)$.

For the Mann-Whitney U test, the test statistic, U, is given by the smaller of U_A and U_B, defined as:

$$U_A = w_A - \frac{n_A(n_A+1)}{2},$$

$$U_B = w_B - \frac{n_B(n_B+1)}{2}.$$

If the p-value is less than 0.01, we usually say that there is very strong evidence against H_0, but everything below 0.05 is considered strong evidence.

2.3.3 Exact p-value by Enumeration

If the sample sizes are small, we can compare two groups of observations using enumeration by utilizing the exact distribution of the test statistic by giving each possible outcome a probability. If we have a null hypothesis that two populations are equal and an alternative hypothesis that the populations are not equal, we can by ranking the observations of the combined population and generating all possible combinations of ranks, find the probability for the observed outcome. By calculating the sum of ranks for each possible rank-combination, we can find the probability of observing the already observed or more extreme [20]. Because we have to find the probability for every possible combination, the number of calculations will quickly get big. The number of possible combinations for one population can be given by the binomial coefficient:

$$\binom{n}{k} = \frac{n!}{k!(n-k)!},$$

where n is the size of the one population, and k is the number of ranks. This means that if the population sizes are, for example, of size nine each, $\binom{9}{18} = 48620$, where 18 is the number of ranks. We then have to find the sum of 48620 combinations. Therefore, the exact method for calculating p-value, should only be used for small sample sizes even though it, in theory, could work for all sample sizes.

If we have two populations of equal length, where one population has ranks: [1, 4, 2], and the other population has ranks: [3, 5, 6], the respective rank-sums are 7 and 14. All combinations of ranks and their sums for one sample is then, as shown in Table 2.2. We can with this information find out how many of these sums are less than or equal to, and bigger than or equal to the first observed sample with the sum of ranks equal to 7, which is 2 and 19 respectively. We want to find the tail set, so because 2 is smaller than 19, the probability of observing ranks where the sum equal to 7 or less is 2/20=0.1. If we want the two-sided test, we can multiply this by 2 and get a p-value of 0.2.

Table 2.2: For calculating p-values by enumeration: All possible rank combinations and respective sums for a combined sample of length 6. Here we only show the ranks for one sample. The other sample's ranks would just be the ranks not present in each row.

Combinations	Sum
1, 2, 3	6
1, 2, 4	7
1, 2, 5	8
1, 2, 6	9
1, 3, 4	8
1, 3, 5	9
1, 3, 6	10
1, 4, 5	10
1, 4, 6	11
1, 5, 6	12
2, 3, 4	9
2, 3, 5	10
2, 3, 6	11
2, 4, 5	11
2, 4, 6	12
2, 5, 6	13
3, 4, 5	12
3, 4, 6	13
3, 5, 6	14
4, 5, 6	15

2.3.4 Multiple Comparisons Problem

Rejecting a null hypothesis because a calculated p-value is less than the chosen significance level does not necessarily mean that the null hypothesis is false. It actually might be true, and the significant result might be due to chance. When testing a set of hypotheses simultaneously, the probability of type I errors increases. For example, if we have 1000 true null hypothesis that we perform tests on with significance level 0.05, we would have to expect 50 of these results to be significant just due to chance. These are 50 incorrect rejections, or type I errors, as previously described. This problem,

that some fraction will be type I errors when doing multiple statistical tests, is called multiple comparisons or multiple testing problem. The goal is, therefore, to reduce the number of false positives. One thing one should have in mind when correcting for multiple comparisons is that the number of false negatives might increase [21]. A false negative is when there is statistical significance, but the test does not detect it.

Controlling the False Discovery Rate with the Benjamini-Hochberg Procedure

Several statistical techniques have been made for preventing incorrect rejections from happening. We will only be looking at one technique: the Benjamini-Hochberg procedure [22]. By controlling the false discovery rate (fdr), i.e., the proportion of significant results that are false positives, using this procedure, we can reduce the number of false positives. Having a set of p-values from multiple hypothesis tests and a chosen false discovery rate, the procedure can be divided into the following steps:

1. sort the p-values in ascending order, $p_1 \leq p_2 \leq ... \leq p_N$;
2. rank the sorted p-values, the smallest p-value with rank $i = 1$, the next smallest with rank $i = 2$, and so forth, where equal p-values get the same rank;
3. compare each p-value to its Benjamini-Hochberg critical value, defined as:
$$H_i = \frac{i}{m}\alpha$$
where i is the rank, m is the total number of p-values, and α is the chosen false discovery rate;
4. reject the null hypothesis for all p-values that are smaller than or equal to the largest p-value that satisfies $p < H_i$, also called the threshold [23].

2.4 Quantile Normalization

Quantile normalization is a statistical technique for normalizing two or more arrays in a set. This can be done either by normalizing the set to a reference distribution, such as the Gaussian distribution or the Poisson distribution or by normalizing the arrays to each other, as we will be doing in this thesis. By sorting each array independently, from smallest to biggest, we can find the mean at each entry, such that the biggest value of all arrays has a mean, the second biggest value has a mean, and so forth. The array of means is then used to replace each entry in the original set at the correct entries so that every array has the same ordering as the original. The short version is that it is transforming the set of arrays to have a common distribution of values. Since we in this thesis will be having the set of arrays stored as a matrix, the algorithm will consist of the following steps:

1. given matrix X of size $p \times n$, where each array is a column;
2. sort each column of X to give X_{sorted};
3. take the means across the rows of X_{sorted} and store this mean to each element at each row to get X'_{sorted}, the mean being the arithmetic mean, defined by the formula:

$$A = \frac{1}{n}\sum_{i=1}^{n} a_i = \frac{a_1 + a_2 + ... + a_n}{n};$$

4. get $X_{normalized}$ by rearranging each column of X'_{sorted} to have the same ordering as the original X [15].

2.5 Data Smoothing

As a part of the results created by our pipeline, some figures containing information about the data obtained during various tasks of the pipeline will be created. Methylation datasets can contain a lot of noise, and we must, therefore, smooth some of the data to be able to present the results. We here present two data smoothing techniques which are utilized in the pipeline.

2.5.1 Centered Moving Average

A moving average is a calculation to get averages of different subsets of a full dataset. Given a list of numbers and a fixed subset size n, we replace the original numbers with the averages of nearby numbers. In this study, we will use a centered moving average, meaning that the averages will be calculated from the subset between $\frac{n}{2}$ numbers to the left and $\frac{n}{2}$ numbers to the right of the original number. If the subset is cut off because of it being at either end of the full dataset, only the numbers available will be a part of the subset used.

2.5.2 One-Dimensional Gaussian Filter

The Gaussian filter is also used for smoothing data, but it uses a different kernel than the moving average smoothing, which uses the mean. The kernel of the Gaussian filter represents the shape of a Gaussian "bell curve," and the one-dimensional Gaussian function is:

$$G(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

where σ (sigma) is the standard deviation of the distribution, and μ is the expected value. We smooth the dataset by convolving it with a Gaussian function. Convolution is the process of multiplying each number of the original list with the neighbor numbers weighted with filter weights from the Gaussian function. The degree of smoothing is determined by the

standard deviation of the Gaussian, such that a larger standard deviation means a larger convolution kernel [24].

Part II

Method and Implementation

Chapter 3

Genetic Data

In biomedical research, the laboratory mouse genome is one of the key information tools for understanding the function of the human genome. Because the protein-coding regions of the mouse are, on average, 85 percent identical to that of the human, a thorough annotation of the mouse genome is of significant value to understanding the content of the human genome. The final pipeline-tool can take both human and mouse datasets as input, and, hence, we will do test runs on both types of datasets. In this chapter, we try to describe what the various input data files are, their format, and how they have been collected.

3.1 Reference genome file

A reference genome is a data structure that represents genetic information for a species. This can be accessed and downloaded from online databases such as UCSC Genome Browser [25], NCBI [26], EMBL-EBI [27], etc. The human reference genome is derived from the sequencing of DNA from numerous anonymous individuals, and the official name for the newest assembly is called Genome Reference Consortium Human Build 38, or GRCh38. In the UCSC Genome Browser, it is referred to as hg38. The most recent mouse reference genome is called GRCm38, or mm10 in the UCSC Genome Browser. In this thesis, we use the refFlat data as reference. The refFlat data files are tab-separated text files containing gene predictions, which are formatted by each row being a gene prediction. What information is stored for each prediction is shown in Table 3.1. In this project, we will only be using the first six of the eleven columns, which are: the gene name as it appears in the Genome Browser, the gene name, the chromosome name, which strand it is, i.e., if the gene is 5′ -> 3′ (+) or 3′ -> 5′ (-), the transcription start site of the gene, and the transcription end site of the gene. One gene name as it appears in the Genome Browser can have several gene names in the refFlat file. Typically, this means that either the transcription start sites differ, or that the transcription end sites differ.

When downloading the refFlat reference data from the UCSC Genome Browser, it will for human contain genes in the 23 chromosome pairs: 22 autosomes and one allosome. The autosome chromosomes are chromo-

somes that are not related to sex, named chromosome 1 to 22. The allosome chromosome pair normally consist of one X and one Y chromosome in females and two X chromosomes in males. These are also called sex chromosomes. Mouse only has 20 chromosome pairs: 19 autosome and the one allosome. The downloaded reference data for both human and mouse additionally contain genes of the mitochondrial chromosome, as well as fix patches, and unfinished and uncertain chromosomes. In this project, we will only use the genes of autosome, allosome, and mitochondrial chromosomes, making the number of chromosomes for human and mouse 25 and 22, respectively. Thus, the rest of the chromosomes not needed in the pipeline must be removed by the user before gene annotation.

Table 3.1: Descriptions of each column in refFlat. We only use the first six columns, i.e., geneName, name, chrom, strand, txStart and txEnd in this project.

Column	Type	Description
geneName	string	Name of gene as it appears in Genome Browser
name	string	Name of gene
chrom	string	Chromosome name
strand	char[1]	+ or - for strand
txStart	uint	Transcription start site
txEnd	uint	Transcription end site
cdsStart	uint	Coding region start
cdsEnd	uint	Coding region end
exonCount	uint	Number of exons
exonStarts	uint[exonCount]	Exon start positions
exonEnds	uint[exonCount]	Exon end positions

3.2 Chromosome sizes file

A file containing chromosome sizes respective to the chromosomes of the reference file is needed in the pipeline. This file is called the genome file, or chromosome sizes file, and can be downloaded from the UCSC Genome Browser for the needed species. It is a tab-separated text file that only contain two columns: chromosome names and chromosome sizes. The chromosome sizes are reported as the number of base pairs. The unwanted chromosomes, as mentioned in the previous section, must also be removed from this data file before being input to the pipeline. Additionally, the file must be sorted, based on the chromosome names, such that the order is as following: $chr1, chr2, ..., chr21, chr22, chrX, chrY, chrM$.

3.3 Tissue-Specific Enhancers file

The tissue-specific enhancer files used in this project are obtained from the paper by Y. Shen et al. [3]. The enhancer files input to the pipeline must be in BED format. The BED format consists of three required fields, or

columns, and several additional optional fields. The first three fields are the chromosome name, the chromosome start site, and the chromosome end site. The optional field is, in the case of the enhancer files input to the pipeline, information on the specific tissue of the enhancers.

3.4 HMST-Seq Sample Data

As previously mentioned, the methylation data for this project is generated by the HMST-Seq method. The general goal for identifying DMRs is to detect regions under certain biological conditions that have different methylation levels when compared to controlled cases [11]. In the current study, we will focus on both 5mC and 5hmC levels. One genetic technique for controlling the conditions of specific genes is called gene knockout (KO). This is a procedure that can be used to silence or inactivate one or multiple genes. Knocking out multiple genes is called double gene knockout (DKO). One reason for knocking out genes is for making a comparison between a KO condition sample and a wild type (WT) condition sample. A WT condition sample is the natural form of a sample, i.e., where there has not occurred any editing or changes of genes. By having one KO condition sample and one WT condition sample, we can, with the analysis-pipeline, possibly detect differences in 5mC and 5hmC regions. One can also use data that does not come from doing controlled gene knockout, but instead having two samples: one test sample and one normal (control) sample, such that the test sample can be from a disease tissue while the control sample can be from a healthy tissue. However, we will, for the most part, refer to test and control samples as KO and WT samples, respectively, in this thesis.

The pipeline can only handle HMST-Seq datasets containing tag counts. This means that the low level analysis, including mapping the raw sequencing tags to a genome sequence and counting the number of tags at each genome site, must already be done before inputting the data to the pipeline. The format of these datasets generated by the HMST-Seq method must be tab-separated text files containing the five columns: chromosome name, genome site, and the tag counts for the three libraries HpaII, BGT and MspI, in that order. The tag counts can, however, be pre-normalized or not. There will be an option in the pipeline-tool for normalizing across the three libraries using quantile normalization.

3.5 BS-Seq Sample Data

Even though the pipeline is primarily developed for handling HMST-Seq datasets, it can be used for doing differential methylation analysis on BS-Seq datasets as well. The datasets sequenced using the BS-Seq method should also be BED-formatted, but where the optional fields are the methylation level and a background level. The background level is not needed in this project and will be removed by the pipeline.

Chapter 4

Software and Hardware

4.1 Programming Language, Libraries, and Software

In this project, we have developed a pipeline working as a command-line tool for analyzing methylation datasets from the HMST-Seq method. Ideally, the pipeline shall have a graphical user interface to make it simpler to use for users, such as biologists, that do not know how to use the command line. However, because of time restrictions, we focused on creating a simple-to-use command-line interface. The pipeline is developed as a Python [28] package that in theory, can be installed on any machine having Python accessible. The pipeline is implemented using Python version 3.6, but test runs are conducted using Python 2.7. The test runs of the pipeline will be described in more detail in Chapter 7.

Python is a high-level, general-purpose programming language, and is a very popular tool for data science and data mining methods. Because it is a general-purpose language, it can be used for doing everything from creating websites to analysis using machine learning techniques. We used Python for this project, because it is a simple language to learn, because of its code readability, and because Python performs very good when doing analysis on large datasets. It comes with a standard library distributed with it. Some of these modules are used in the pipeline, such as argparse and multiprocessing. The module called argparse is used to make a user-friendly command-line interface. It helps with generating help and usage messages automatically, and issues errors when the program is given invalid arguments. For parallelization of some of the most computationally heavy tasks in the pipeline, we have used the multiprocessing module, which allows Python to use multiple processors on the given machine. Other modules that are distributed as a part of the standard Python library are also utilized. Further Python packages which are not in the standard library are used in the pipeline as well. This includes pandas [29], NumPy [30], Matplotlib [31], seaborn [32], and SciPy [33]. These can be installed using a package manager such as pip or conda [34]. The pandas package is a data analysis library which is heavily used in the implementation of the pipeline. The package offers in-memory data structures as well as a broad set of operations for manipulating datasets. The data manipulation operations of the module make it easy to do operations on whole data structures at the same time. It also makes the reading and writing of the data between the in-memory data structures and different file formats very

simple. Matplotlib and seaborn are data visualization packages used for plotting the results, while NumPy is the fundamental package for scientific computing in Python. To be able to install the pipeline itself, we utilize a Python library called setuptools, which is a system for developing software packages in Python. It allows us to have a setup.py script for easily building the analysis pipeline package so that it can be installed using the following command:

```
$ python setup.py install
```

Besides packages and modules related to Python, we utilize software in the pipeline that is outside of the Python library. BEDTools [35] is one of them. BEDTools is a library of command-line tools for a wide-range of genomics analysis tasks. It is available for installment through various package managers such as conda. There is also an option for using MATLAB [36] in the Python pipeline. This is done through a MATLAB engine API which can be installed as a Python package. MATLAB is a numerical computing environment and proprietary programming language. Proprietary, meaning that it is a non-free, closed source software. Because MATLAB is proprietary, users of the pipeline might not have access to it. Hence, R [37] can be used for the same task MATLAB is used. R is a software environment for statistical computing and graphics that is freely available under the GNU General Public License. What features of BEDTools, MATLAB, and R we use, for what purpose, the necessity, and why, is discussed in more detail in the next two chapters.

4.2 Hardware for Testing

The test run conducted in this thesis was performed on the Abel computer cluster - a high performance computing resource at the University of Oslo, hosted at USIT by the Research Infrastructure Services group. The Abel cluster is comprised of more than 650 compute nodes, each running on dual Intel E5-2670 (Sandy Bridge) with 2.6 GHz, yielding 16 physical computer nodes, and 64 GB Samsung DDR3 memory operating at 1600 MHz. All compute nodes are running Linux, 64 bit Centos 6. Abel uses the queue system: Slurm Workload Manager, to ensure effective utilization of the Abel infrastructure.

Chapter 5

The Pipeline Tasks and Flow

There are already multiple tools available for analyzing DNA methylation data. Many of them are made for data generated from bisulfite sequencing (BS-Seq). Since BS-Seq does not differentiate 5mC and 5hmC, we cannot use these tools for the current project. There are also tools developed for analyzing both 5mC and 5hmC data, but it is challenging to use these existing tools to analyze both datasets simultaneously. Thus, a new computational pipeline considering a proper statistical method for analyzing DNA methylation data produced by the HMST-Seq method has been made: *HMST-Seq-Analyzer*. Even though the main focus in the current thesis will be differential analysis of 5mC and 5hmC data, the pipeline will, as a bonus, also be able to do differential analysis on datasets generated by the BS-Seq method. In this chapter, we will give an overview of the pipeline, and the various tasks of the pipeline, where the focus will be on what each task does. In the next chapter, we will look more into the architecture of the pipeline, and the inputs and outputs of the various tasks.

5.1 The Pipeline

High level, the pipeline consists of eight tasks that must be run separately and in a certain order to function as intended. The pipeline's eight tasks and main flow is shown in Figure 5.1. In this figure, we can see the names of the different tasks and the order they are meant to be run in from start to finish. The first two tasks, **Data Preprocessing** and **Gene Annotation**, can be run regardless of the order because they are not dependent on each other. After these two steps, the running order shall be: **Find MRs**, **Preparation for DMR Search**, **DMR Search**, **Plot Preparation**, **Plotting**, and **Clean Folder**. The first seven tasks have to be run in order to produce all possible results such as DMR findings, and all of the results for figures. The eighth task does not influence the final results, but is for removing temporary files from the output folder that are created during the pipeline-run. In this way, by having a pipeline divided into multiple tasks, users can have an opportunity for experimental variation.

As previously mentioned, the pipeline is developed in Python. An overview of the package folder, called HMST-Seq-Analyzer, is

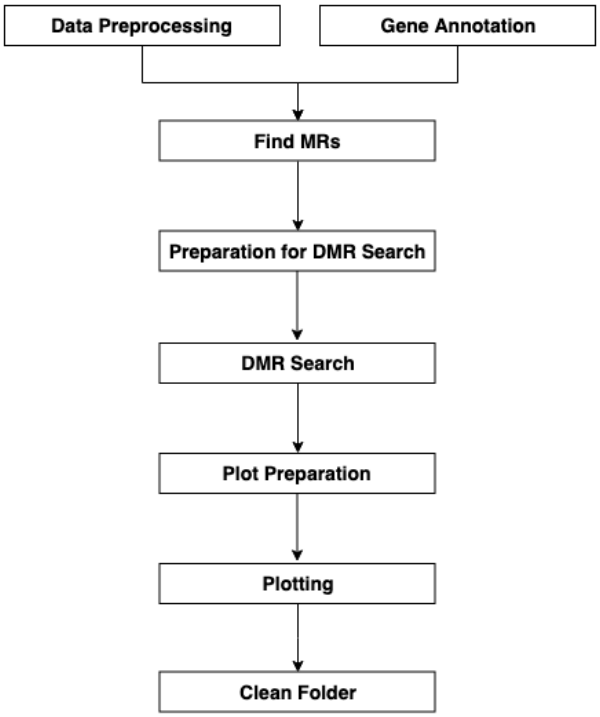

Figure 5.1: The eight pipeline-tasks of the Python pipeline, and the order they should be run in.

shown in Figure 5.2. Within this folder, there is another folder, hmst_seq_analyzer, which contains the scripts of the pipeline. The main file, hmst_seq_analyzer.py, is the Python script that is called every time a user interacts with the pipeline. When a user wants to run a pipeline-task, hmst_seq_analyzer.py is called, with the input argument being the name of the task. When the pipeline is successfully installed as described in section 4.1, the name of the pipeline is hmst_seq_analyzer, and users can interact with it using the hmst_seq_analyzer command. By entering *hmst_seq_analyzer -h* in the command line, a help message showing what the names of the various tasks are, a short task description, as well as the usage of the pipeline will be shown. This help message can be seen in Figure 5.3. In addition to the main pipeline file, the hmst_seq_analyzer folder contains the folder called scripts_high, where scripts of each separate task will be found. The pipeline consists of 14 functions, spread across the first seven tasks. These functions are in the folder called scripts.

The readme file contains essential information for users of the package, such as how to install the package, how to install the requirements, a demo run, as well as some general information that can be useful for a user to know. The file named requirements.txt contains names of Python packages and software that must be installed by the users. setup.py is the file allowing users to install the Python package. When the pipeline, the required packages, and software has been installed, users can utilize the pipeline package from anywhere, meaning that it can be run from whichever directory the user wants to.

```
HMST-Seq-Analyzer
|-- hmst_seq_analyzer
|   |-- hmst_seq_analyzer.py
|   `-- scripts_high
|       `-- scripts
|-- readme
|-- requirements.txt
|-- runner.py
`-- setup.py
```

Figure 5.2: The HMST-Seq-Analyzer Python package directory tree. The files of the two folders scripts_high and scripts are not shown, but these are the folders containing the different Python files for all the tasks of the pipeline.

5.2 Gene Annotation

In order to find differentially methylated and hydroxymethylated regions (DMRs) in the methylation data, we first need to extract genomic regions from the reference genome file, refFlat. The reference genome represents a set of genes for a species, which for this project are either mouse or human, as described in section 3.1. The genomic regions we extract from each gene during this task are transcription start site (TSS), gene body, transcription end site (TES), 5′ untranslated region (5′UTR), and intergenic. These genomic regions will in the **Find MRs** task be used to assign the methylated sites of the sequencing data to. TSS and TES are considered sites and not regions, but we will, in this thesis, consider them as regions. This is because we are interested in the regions around the TSSs and TESs. By doing gene annotation on the reference genome, we can identify locations for all such genomic regions.

Before the gene annotation, the reference is cleaned. Cleaning the reference includes removing duplicates and specific genes called microRNA genes. MicroRNAs are non-coding RNAs that regulate the gene expression negatively by inhibiting transcription from occurring [38]. These have gene names starting with the text "MIR" in the refFlat files. Users of the pipeline can decide whether such genes shall be removed or not. We will get back to what the user options for each task of the pipeline are in the next chapter. As mentioned in section 3.1, the refFlat file can have more than one of the same gene name, but that has different transcription start or end sites. When extracting regions such as TSS and TES, this could result in there being multiple of the same region. Therefore, we merge such regions into one unique. For example, if two genes have the same TSS but different TES, we would after gene annotation have two of the same TSS region, which is not wanted.

The length of the different genomic regions, such as TSS, gene body, TES, 5′UTR, and intergenic, can be influenced according to a user's preferences. Figure 5.4 illustrates how these five genomic regions are extracted from the genes of a reference genome, i.e., where the extracted genomic regions are located relative to each other. If a user plans to use the same genomic region sizes across multiple experiments with the same species, this task only needs to be done once.

```
usage: hmst_seq_analyzer <task> [<args>]

Tasks available for using:
   gene_annotation            Cleans reference file and creates genomic region files
                              (TSS, geneBody, TES, 5dist and intergenic) from the reference

   data_preprocessing         Creation of 5mC and 5hmC files, quantile normalization

   find_MRs                   Extracts genomic regions from 5mC/5hmC-files and finds
                              methylated regions

   prepare_for_DMR_finding    Finds overlapping methylated regions between MRs in
                              WT condition samples and KO condition samples

   DMR_search                 Finds differentially methylated regions

   prep4plot                  Prepares files for plotting

   plot_all                   Plots hyper versus hypo differentially methylated regions,
                              enhancer methylated regions, TSS_gene_TES methylated regions
                              and relative density of significantly modified sites in MRs
                              with versus all sites in MRs

   clean_files                Removes some unwanted files. Please only use after prep4plot
                              is already done

HMST-Seq Analyzer

positional arguments:
  task        Pipeline task to run

optional arguments:
  -h, --help  show this help message and exit
```

Figure 5.3: The help message for hmst_seq_analyzer. It shows the eight tasks of the pipeline, a small description of each task, and how to run the tasks. By further entering *hmst_seq_analyzer <task> -h*, where <*task*> is the name of the task, a help message for the specified task is shown.

5.3 Data Preprocessing

The task called **Data Preprocessing** consists of two parts. The first part is applying normalization to the tag counts among the three libraries HpaII, BGT, and MspI, of the HMST-Seq datasets. The normalized libraries are then used in the second part of this task, which is calculating the relative abundance of 5mC and 5hmC. The abundance of 5mC can be determined as the ratio between the tag counts in the libraries BGT and HpaII, while the abundance of 5hmC can be determined as the ratio between tag counts in the MspI and BGT libraries [8]. This is done by simply dividing the tags in BGT by the tags in HpaII for 5mC, and dividing the tags in MspI by the tags in BGT for 5hmC, at each site. These new numbers are what we call 5mC and 5hmC levels in this thesis.

5.4 Finding MRs and DMRs

The process of finding DMRs and DhMRs in methylation data includes finding the genomic distribution of 5mC and 5hmC and comparing methylated genomic regions across different biological conditions. **Find MRs**, **Preparation for DMR Search**, and **DMR Search** are the three tasks doing exactly this. These are the three most computationally heavy tasks in the pipeline. They are closely related, and thus, we present these tasks

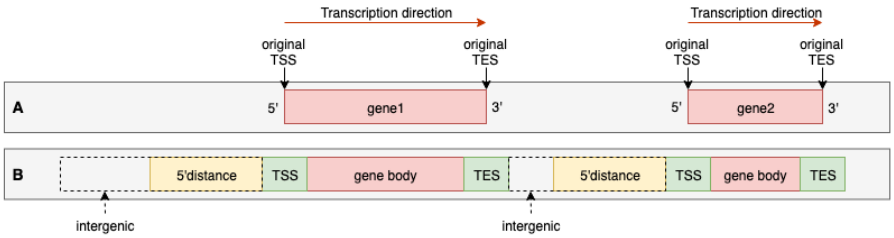

Figure 5.4: The different genomic regions extracted from the genes of the reference genome, and where they are located relative to each other. The extracted genomic regions are TSS, gene body, TES, intergenic, and 5' UTR. **A** shows two genes of the reference genome, while **B** shows the extracted genomic regions. Both genes have + strands.

under the same section.

5.4.1 Find MRs

The first of the three tasks related to finding DMRs/DhMRs is called **Find MRs**. During this task, we find the genomic distribution of 5mC and 5hmC, as well as defining 5mC and 5hmC regions, called methylated and hydroxymethylated regions (MRs and hMRs). To make things simpler, we will from now on refer to both MRs and hMRs as MRs.

An MR can be defined as a cluster of 5mC or 5hmC sites in one genomic region of a gene from the reference. To find MRs, we must first align the 5mC/5hmC sites to the genomic regions of the reference. This can be done by mapping the 5mC/5hmC sites to the genomic regions, or by identifying overlaps between the genomic regions and 5mC/5hmC sites using the BEDTools feature called intersect. The intersect feature is one of the BEDTools tools, and it allows a user to look for overlaps between two sets of genomic data, which is why we chose to use this for the mentioned mapping. When we run BEDTools intersect between the two sets: one genomic region and either one of 5mC/5hmC set of a condition sample, we will without specifying any options get the shared interval between the sets. This means we cannot tell from which each intersection came. However, if we add the options *-wa* and *-wb* when running the intersection command, both the shared interval and the regions of the first set from which the shared interval comes from, will be in the output. This is illustrated in Figure 5.5. In the figure we have two sets of genomic regions, A and B, that, when A intersect B is run with the *-wa* and *-wb* options will give both the shared interval and the regions from A. The output of this will be all genomic regions and the 5mC/5hmC sites within the start and end site of the genomic regions. However, BEDTools does not give us MRs when finding overlaps. That will be the next step.

As previously mentioned, an MR is a cluster of 5mC or 5hmC sites in a gene's specific genomic region. However, in this pipeline, the cluster must fulfill two requirements to be considered as an MR: (1) the cluster must have at least N 5mC or 5hmC sites, and (2) there can not be more than a bps

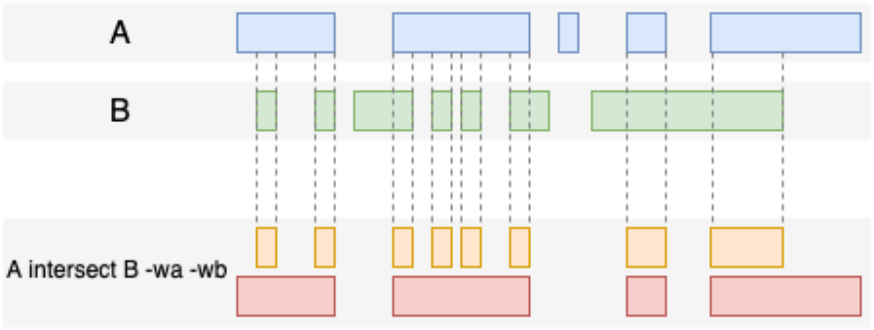

Figure 5.5: An example of BEDTools intersect where A and B are two sets of genomic regions. If we run A intersect B with the options *-wa* and *-wb*, we get both the shared interval (yellow) and the regions of A from which the shared interval came from (red), as seen in the bottom grey area.

between each adjacent site in the cluster, which is ordered by the genome sites. This means that if we have a cluster of length 5 and N is equal to 5, but the inter-distance between the two sites in the middle of the cluster is more than the specified a, it will be discarded, i.e., not be considered as an MR. A simple Python script illustrating this logic is shown in Figure 5.6. In this code, the variable *possible_MRs* is an ordered list containing methylated sites overlapping with one genomic region. For example, if the genomic region is the leftmost red block in Figure 5.5, the two yellow blocks overlapping with it are the clusters of methylated sites. The goal is to find out if these two clusters are MRs or not. If they are MRs, they will, in the Python code in Figure 5.6, be included in the list called *MRs*. If a cluster of sites is split because adjacent sites are having inter-distance longer than a, both parts of the split can still be MRs if the requirements for each part being an MR, are fulfilled. This part does not separate each MR of such a split but finds all sites that are included in MRs. In the next task, each MR is separated from each other.

5.4.2 Preparation for DMR Search

While finding MRs is done for one sample at a time, finding DMRs is on the other hand done by comparing two samples, such as a WT condition sample and a KO condition sample, as described in section 3.4. Before we can find DMRs using a statistical hypothesis test, we must first find overlapping MRs between two samples. This is done in the task of the pipeline called **Preparation for DMR Search**. In this task, we use IDs to find overlapping MRs. An ID is a unique combination of chromosome name, gene name as at appears in the Genome Browser, gene name, strand, start and end position of this genomic region, and original transcription start and end sites of the gene from the reference genome. It is used such that the same genomic regions can be found between the samples. By finding all unique IDs from the combined data of KO and WT condition samples, we could extract all 5mC/5hmC sites from both

```
tmp = []
MRs = []
for site in possible_MRs:
  if len(tmp) == 0:
    tmp.append(site)
  else:
    if site - tmp[-1] <= a:
      #length is less than a
      tmp.append(site)
    else:
      if len(tmp) >= N:
        #MR found
        MRs.append(tmp)
      tmp = [site]
if len(tmp) >= N:
  #MR found
  MRs.append(tmp)
```

Figure 5.6: Simplified Python script for finding methylated regions from the ordered list of methylated sites called *possible_MRs*.

samples. However, we are not interested in MRs of WT data that are not present in KO data, and can, therefore, extract all 5mC/5hmC sites for both KO and WT data from the unique IDs of the KO data alone. The way this is done is by doing a full outer join on the 5mC/5hmC sites of the unique IDs. A full outer join gives all 5mC/5hmC sites in MRs from both KO data and WT data whether the sites are in common or not.

Missing Data

Missing data due to experimental and technical noise is quite common in datasets obtained through high-throughput sequencing. Hence, some 5mC/5hmC sites may not be observed during the sequencing experiment. This means that not all 5mC/5hmC sites in an overlapping MR are in common, but can have one or many missing sites from both data conditions. An illustration showing an overlapping MR where both KO and WT data have one missing site each is shown in Figure 5.7. Missing data can, in general, be handled in three different ways: by imputation, by removing the affected data, or by continuing with the data as it is. By imputation, we mean replacing the missing entries by some value, like the mean, median, or simply just zeros. One argument for replacing the missing values of overlapping MRs by zeros in this project is that it is similar to the real situation. Because, when a site is not observed, it means it is not present in the experiment, and a methylated level of zero resembles that. Another way of imputing missing values is by interpolation, but we will not have that option in this project because it might add noise to the data. If we were to discard overlapping MRs having missing values, we would lose a lot of possible important data. Therefore, discarding overlapping MRs missing values would not be appropriate. The last method for handling missing data is to leave the data untouched. The only

problem with this is that it is not certain that the test method or function we will use for the comparison of the two sets of 5mC/5hmC sites can handle non-existing numbers or NaNs in Python. NaN means "not a number" and is a Python float that cannot be expressed as a number.

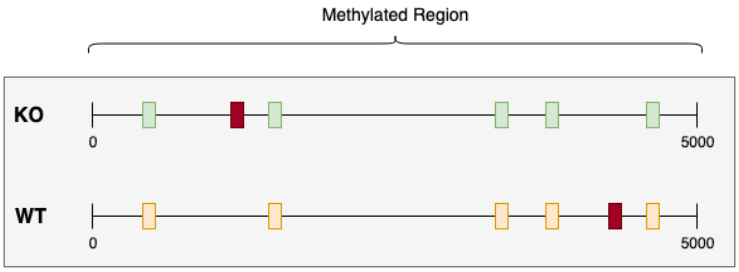

Figure 5.7: One MR, where the green and yellow sites represents overlapping methylated sites, and red represents missing values. The missing data can for example be handled by imputation, e.g., by filling in the missing values by for example the median or zeros.

The 5mC/5hmC sites of the overlapping MRs might not fulfill the requirements to be considered MRs. Therefore, we validate each overlapping MR and its 5mC/5hmC sites, as previously described and shown in Figure 5.6, before going to the next step: finding DMRs and DhMRs. This validation is also done so that each MR is one row in the output, such that if a split occurs, as mentioned in section 5.4.1, within one genomic region, each part is on separate rows.

5.4.3 DMR Search

Differentially methylated and hydroxymethylated regions (DMRs and DhMRs), are genomic regions having different methylation status across biological samples. When testing if an MR has different 5mC/5hmC levels across two samples, a proper statistical hypothesis test should be used. In the article by Y. Xia et al. [8], the authors found DhMRs and DMRs in H9 human embryonic stem cells (hESCs) by using a Wilcoxon rank-sum test with the p-value less than 0.05 to define DhMRs and DMRs. In F. Gao et al. [10] they also used a Wilcoxon rank-sum test with the p-value less than 0.05, to specify DMRs and DhMRs between hepatocellular carcinoma (HCC) and non-HCC samples, sequenced with HMST-Seq. Since both Y. Xia et al. [8] and F. Gao et al. [10] used Wilcoxon rank-sum test to find DMRs and DhMRs in HMST-Seq data, it encouraged us to apply the same method for the present project. In section 2.3.2, we described the basics of such a test. For simplicity, we will mostly use DMRs when referring to both DhMRs as DMRs.

The biological samples should already be overlapping, and missing sites should be handled accordingly, as described in the previous section, which means we can now detect DMRs by comparing 5mC/5hmC levels of the two samples in overlapping MRs. The pipeline will have an option for users to choose between a few different approaches for detecting DMRs,

which we will get back to in the next chapter. After running a statistical hypothesis test on all overlapping MRs, each of the tests will give a corresponding p-value. Users can then chose to reject all hypotheses having p-value less than a chosen α, or run a Benjamini-Hochberg correction. The MRs that are rejected are then considered differentially methylated. A DMR can be either of the two types: hypomethylated or hypermethylated. If a DMR is hypomethylated, it means there is a decrease in the methylation levels, i.e., that the KO condition sample is less methylated than the WT condition sample in this specific DMR. If, on the other hand, a DMR is hypermethylated, it means there is an increase in methylation levels. The way we measure if there is an increase or decrease in methylation levels in a DMR is by finding the relative ratio [39] (rratio) between the two. The rratio is given by the following formula:

$$rratio = \frac{\mu_{KO} - \mu_{WT}}{(\frac{\mu_{KO} + \mu_{WT}}{2})},$$

where μ_{KO} is the median of the unimputed methylated levels in the KO data, and μ_{WT} is the median of the unimputed methylated levels in the WT data. If rratio is above 0, the DMR is considered hypermethylated. If it is below 0, the DMR is considered hypomethylated.

5.5 Plot Preparation and Plotting

In addition to finding DMRs, the pipeline stores different information about the methylation data that is obtained during a run of the various tasks. In the next chapter, we will show where the different information comes from, i.e., from which task. To visualize this information, the pipeline creates four different kinds of figures. In this section, we will describe the different figures that are created and how they are made. In chapter 7, we will do test runs of the pipeline, where all figures produced by the pipeline will be shown.

Since the mentioned tasks are closely related, we will present the different kinds of figures that are made by the pipeline instead of the two separate tasks, **Plot Preparation** and **Plotting**. In the next chapter, we will show the input arguments for these tasks separately.

5.5.1 Plotting DMR and DhMR Distribution

To present the distribution of DMRs and DhMRs over the various genomic regions we plot a figure showing the number of DMRs as percentages of the total number of overlapping MRs in the specified genomic region in a bar chart. In this figure, we differ between hypomethylated and hypermethylated DMRs, where hypermethylation is an increase in the methylation level, and hypomethylation is a decrease. Hence, the figure will contain four separate subplots, each showing the distribution across genomic regions. The four plots shows hyper-DMRs, hypo-DMRs, hyper-DhMRs, and hypo-DhMRs as percentages of overlapping MRs.

5.5.2 Plotting Relative Density for MRs

The next figure that is created in the pipeline shows the distribution of significantly modified MR sites across the various genomic regions. This figure is presented as a bar chart as well, showing the ratio of significantly modified MR sites against all MR sites. A significantly modified site can be defined as a methylated site (5mC or 5hmC), having methylation level above a certain threshold, such as 1, which they used in Y. Xia et al. [8]. This figure will contain two subplots, one for 5mC distribution, and one for 5hmC distribution.

5.5.3 Plotting Distribution of 5mC/5hmC Levels of MRs in TSS, Gene Body, TES and Enhancer Regions

As mentioned in section 5.2, each genomic region such as TSS, gene body, TES, 5'-UTR, and intergenic, is calculated based on the gene annotation in the reference genome. In the pipeline, the length of every gene's TSS and TES region is the same across all samples. Thus, we can generate figures that show the distribution of 5mC and 5hmC levels of MRs around the genomic regions, such as TSS and TES. We are also interested in the distribution of 5mC and 5hmC levels of MRs in the gene body region, even though different genes have different lengths. Thus, we create a combined figure showing the distribution of methylation levels of TSS, gene body, and TES regions, for both 5mC and 5hmC. Additionally, the distribution of 5mC and 5hmC levels of MRs in enhancer regions are also illustrated.

To plot the distribution of 5mC and 5hmC MRs for one of the genomic regions, we have made an approach for computing the average methylation level of each site in a set of MRs. For all MRs of one sample in a given genomic region, the transformation is divided into the following four steps:

1. The first step is to map the methylated sites of each MR to the same range. Because each MR in TSS and TES region have the same length, the methylated sites of these two regions can be mapped easily to the same range. However, gene body MRs have different lengths and must be mapped to a common range. We do this by normalizing and scaling the methylated site to be in a new range. We can then find the number y in the new range $[c, d]$, from a number x in the old range $[a, b]$. The formula for this transformation is:

   $$y = \frac{x - a}{b - a}(d - c) + c$$

   For example, if we have a methylated site at genome site 100 in an MR with start and end positions 0 and 200 and we want to transform this site to the range $[1000, 2000]$, the translated number is 1500. This translation is done for all MR sites.

2. Normally, the span of an MR will be anywhere from a couple of hundred to hundreds of thousands of base pairs long. The number

of methylated sites in each MR can also vary a lot, depending on the sequenced samples. If we have many MRs with few methylated sites but long spans, we have a very sparse data set. Therefore, we shall increase the number of methylated sites in each MR. For example, we can increase the number of methylated sites using data interpolation, which is a method to approximate the data distribution based on a function such as $y = f(x)$. We can then use this function to find values for new data points, which, in our case, will be the methylated sites. There are many different types of interpolation algorithms such as spline and linear interpolation, and we apply a one-dimensional nearest neighbor algorithm in this pipeline. The nearest-neighbor interpolation chooses the nearest value for a new data point. By using this algorithm, we are able to obtain new data points and increased data size based on the distribution of the input data, and hence making the data denser. In our case, we are estimating new methylation sites between the first and last methylated site in an MR, with a specific step size. In Figure 5.8, the estimated methylated sites from the nearest-neighbor interpolation method are illustrated using a simple example, where we show the data before step 1, after step 1, and after step 2 of the data transformation.

3. When the data has been increased by interpolation, we can take the mean of all methylation levels at each site of the new span, across all MRs in the genomic region. The resulting data contains methylation level-averages in the new span.

4. If we plot the methylation level-averages, it will contain a large amount of noise and no informative figure will be produced. Therefore, we apply a simple centered moving average method, as discussed in section 2.5.1, to smooth the averages, giving a better and more informative figure.

Since the genomic regions TSS, gene body, and TES are very closely related, the obtained averages of these regions are plotted together in the same figure, by concatenating the three data sets in the above order. However, if one or more of these regions does not contain any MRs, each genomic region will be plotted separately. Enhancer regions will be plotted in separate figures, regardless.

After applying the centered moving average, a one-dimensional Gaussian filter, as described in section 2.5.2, is further applied to smooth the remaining noise of the data. In Figure 5.9 and Figure 5.10, we illustrate the three stages of the data: raw averages, after applying the centered moving average, and after further applying a one-dimensional Gaussian filter. We see in these figures that there is much noise in the raw average data. If the trends of the data should be shown, the moving average smoothing is necessary. There is still some small noise after the average smoothing, which is decreased as shown in the latter figure.

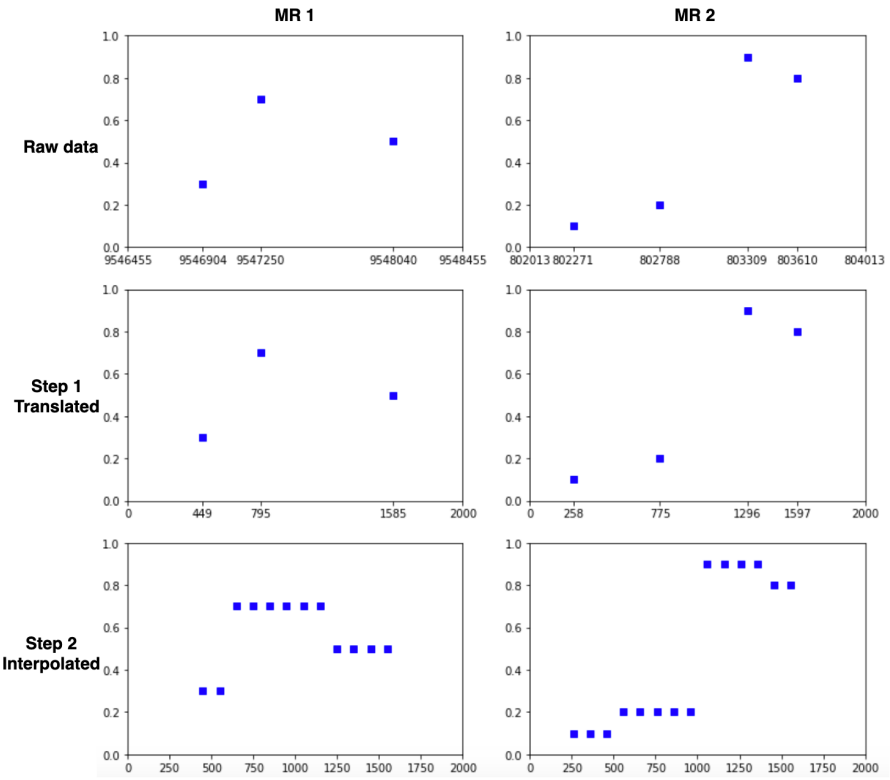

Figure 5.8: The first two steps of transforming the raw methylation levels of MRs for plotting the methylation distribution. The x-axes are showing the genome sites, and the y-axes are showing the methylation levels. The first row is the raw data of two different MRs. The second row is step 1, where the raw data has been translated to be in the span between 0 and 2000. In the third row, we see step 2, where the translated data have been increased by nearest-neighbor interpolation, with a step size of 100. After the second step, we take the average at each site over all MRs and smooth the data by the centered moving average method.

5.6 Clean Folder

The last task of the pipeline is a simple deletion of temporarily created files that are not needed after the figures have been plotted. The files that are removed have been created by the two tasks **Data Preprocessing** and **Find MRs**. This is not an important task but can be of some help to make the output folder of the pipeline cleaner.

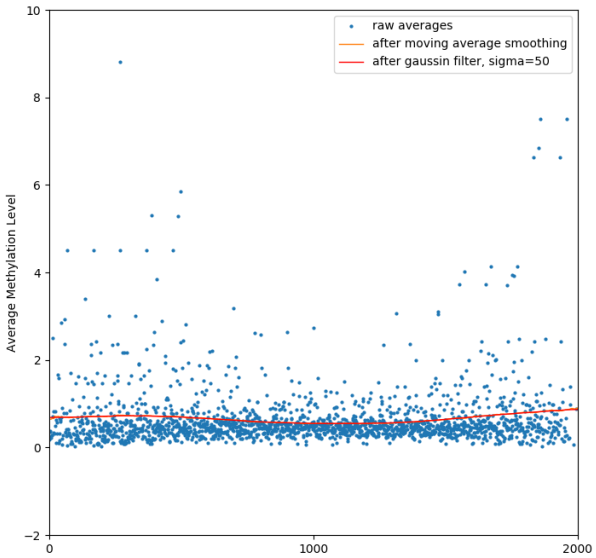

Figure 5.9: An example of all the three stages of the methylation level-averages data: raw averages, after centered moving average smoothing (subset size=500), and after further smoothing using a one-dimensional Gaussian filter with sigma=50. The y-axis is set to -2 and 10 to illustrate how the centered moving average smoothing fits the raw averages.

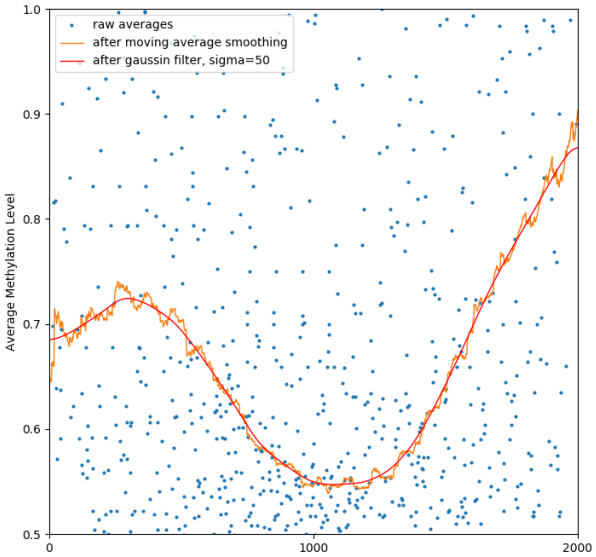

Figure 5.10: The same data as in Figure 5.9, but where we have limited the y-axis to 0.5 and 1 to better illustrate the noise before and after the one-dimensional Gaussian filter.

Chapter 6

Pipeline Architecture and User Interaction

In the previous chapter, we have described the workflow of the pipeline and the first seven tasks in detail. Here, we will look at the architecture of the pipeline: how the different tasks interact with each other, the input arguments and their role in the pipeline, the input files, and the output files.

The pipeline consists of 14 main functions, which is spread across the first seven tasks. The seven task scripts do not perform any of the computations but call the low-level functions with a combination of various input files and a set of input arguments. The architectural design of the pipeline is shown in Figure 6.1, where the first seven tasks are in green and the 14 main functions inside the respective tasks are in red. The pipeline has three primary input datasets that must be provided by the user. They are shown in grey such as the refFlat file, the chromosome sizes file, and the methylation datasets. Also, users can input a tissue-specific enhancer file, but it is not required. Output files from various tasks are shown in blue and purple. The blue-colored datasets are the main data files that are needed throughout the pipeline, while the purple files are used for the final plotting. With each of the blue datasets, comes a separate file containing the file names and file paths for all files of the respective dataset. We call these separate files "list" files. By having these "in-between"-files users can input the list file containing the file names of a dataset instead of inputting each of the dataset files separately. This makes the pipeline more manageable for the user. In Figure 6.2, we illustrate this idea by using the output file sets from the **Find MRs** task. Users can then input these two list files, shown in the figure as: "MRs files KO list" and "MRs files WT list", for KO and WT data, respectively, to the task **Preparation for DMR Search**. All the main data files created in the pipeline (blue) are stored in a folder named data inside the main output folder, while the list files are stored directly in the main output folder. This avoids data files getting mixed up with the list files.

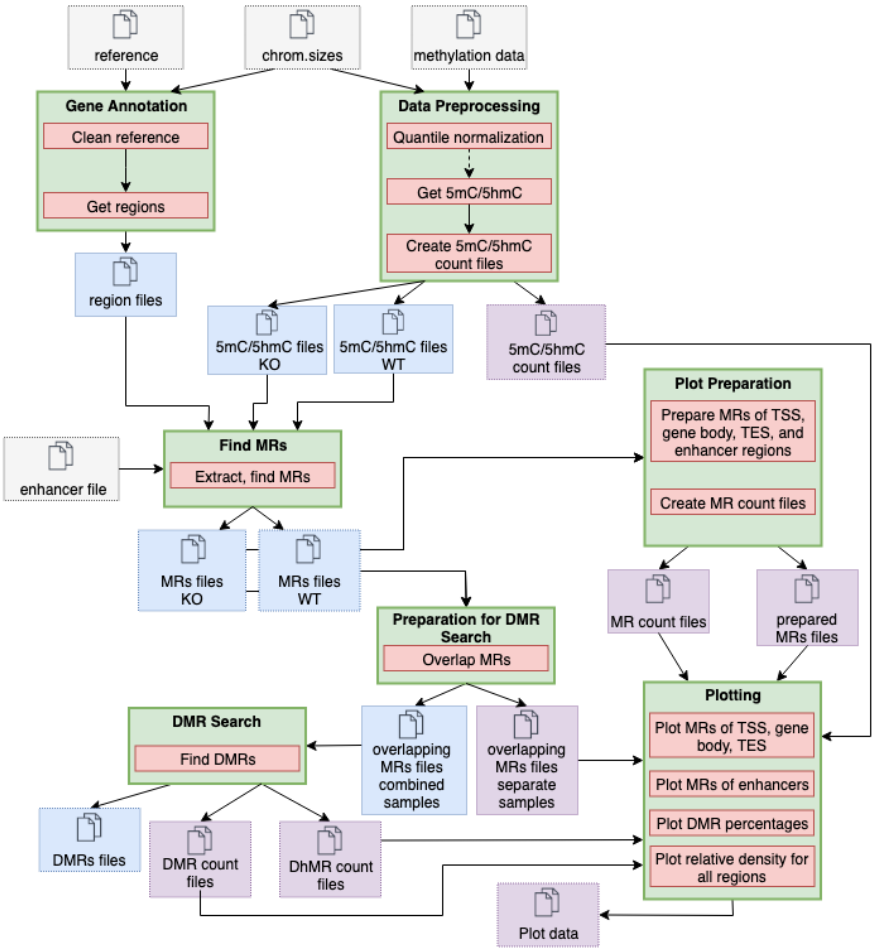

Figure 6.1: The pipeline architectural design. The seven first tasks of the pipeline are shown in green, containing the low level main functions in red. The main input data files that the user must provide to the pipeline are shown in grey. The blue colored datasets are the main datasets needed throughout the pipeline, while the purple datasets are used for the final figure plotting.

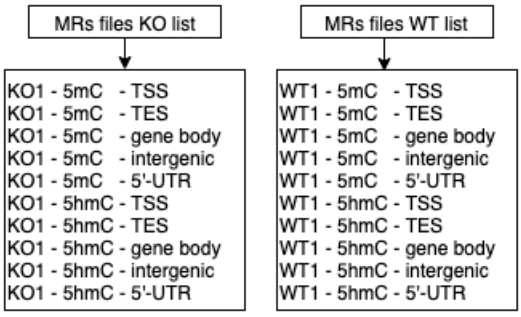

Figure 6.2: Illustrating the list files using the two output datasets of the task **Find MRs**, and the two list files containing all file names of the respective dataset. Here, we assume there is no available enhancer file.

6.1 Gene Annotation

As mentioned, by entering hmst_seq_analyzer followed by a task name and the *-h* argument in the command line, more detailed information will be shown, such as the usage of the task, what the input files, and input arguments are. For example, entering *hmst_seq_analyzer gene_annotation -h* in the command line, a help message for the gene_annotation task will be displayed. As mentioned earlier, some input files, such as the reference genome file and the chromosome sizes file are required. Besides the required arguments, a few optional arguments are needed too, which have default values. For the **Gene Annotation** task, all arguments are shown in Table 6.1, where the first four arguments are required. The two files, refFlat and chromosome sizes file, are input to this task by the arguments *-r* and *-g*, respectively. The user also needs to tell whether the methylation data comes from human or mouse. The human genome contains 25 chromosomes while the mouse genome contains 22 chromosomes, as discussed in section 3.1. This is specified by the argument *-hu*. If the argument *-n* is *no*, the chromosome sizes file has non-numerical chromosome names, and the output data will also have non-numerical chromosome names. However, if it is *yes*, then the output data will have numerical chromosome names (e.g., chromosomes X, Y, and M are changed to 23, 24, and 25 for human, and 20, 21, and 22 for mouse, respectively).

The optional arguments *-X* and *-Y* are related to calculating the genomic regions TSS, TES, and gene body. For a gene, the TSS region starts X bps upstream from the transcription start site, and ends Y bps downstream from the transcription start site. The TES region starts Y bps upstream of transcription end site, and ends X bps downstream of transcription end site. Thus, the gene body regions are contracted by Y in both directions. The optional arguments *-M* and *-N* specify the start and end of 5′-UTR regions, where both numbers are upstream of the genes transcription start site (N>M). The default values of the input arguments X, Y, M, and N have been chosen based on the paper by B. Tang et al. [40].

If the argument *-i* is *yes*, then intergenic regions will be defined between the raw genes (i.e., excluding the regions inside the reference's transcription start and end sites). If *-i* is *no*, then the TSS and TES regions will also be excluded from the intergenic regions. For an intergenic region to be included in the output dataset, it has to be of a certain length. By using the argument *-l*, users can specify the intergenic region's minimum bp length. There are some genes in the reference that are quite short. If X and Y are big enough, then the lengths of some gene body regions will be less than zero. These gene body regions will not be included in the output dataset. However, a user still might want to keep the TSS, TES and 5′-UTR regions of such genes. Hence, if option *-rem* is set to *no*, then TSS, TES and 5′-UTR are kept. If it is set to *yes*, then such genes will not be included in the output dataset at all. All output data of each task can be stored in a specific folder based on a user's selection by using the optional argument *-F*. The default output folder name is *out*.

The gene annotation task's primary goal is to find TSS, gene body, TES,

Table 6.1: The **Gene Annotation** task input arguments, both required (first four), and optional.

Default	Argument	Description
None	*-r/--referenceFile*	reference genome file
	-g/--genomeFile	chromosome sizes file
	-hu/--human	*yes* if sample is from human, *no* if mouse
	-n/--numericalChr	*yes* if the chromosome names of the chromosome sizes file is numerical, *no* if not
yes	*-rm/--removeMir*	*yes* if gene names starting with "MIR" should be removed, *no* if not
1000	*-X*	the number of upstream bp to the transcription start/end site
1000	*-Y*	the number of downstream bp from the transcription start/end site
10000	*-M*	the number of upstream bp from the transcription start site
1000000	*-N*	the number of upstream bp from the transcription start site (N>M)
yes	*-i/--intergenicBTGenes*	*yes* if intergenic regions should exclude only original reference genes, *no* if TSS and TES should be excluded as well
2000	*-l/--minIntergenicLen*	minimum number of bp in intergenic region to be included
yes	*-rem/--removeShort*	*yes* if TSS, TES and 5'-UTR should not be included when gene body size is less than zero, *no* if they should be included
out	*-F/--folderOut*	the name of the output folder

5'-UTR, and intergenic regions based on the genes of the reference file. Hence, the output consists of five BED-formatted files, one for each of the five genomic regions. The five file names are also put in a separate list file. In addition, a filtered and BED-formatted reference file will also be an included output of this task, cleaned as described in section 5.2.

6.2 Data Preprocessing

All input arguments for the **Data Preprocessing** task is shown in Table 6.2. The main function of this task is to preprocess the data of the methylation files. Therefore, the two main inputs are the KO and WT condition sample data files, which are presented to the pipeline by the two arguments *-fko* and *-fwt*. The pipeline can handle one or more of each condition sample by adding each argument multiple times. Since the pipeline can be applied to methylation data produced by both bisulfite sequencing (BS-Seq) and HMST-Seq, the input argument *-m* can be used to distinguish the two: *yes* implies the input data already contains methylation-levels (BS-Seq), and *no* means the data contains the HMST-Seq library tag counts. The computation of 5mC/5hmC levels is not needed for the BS-Seq data because it does not contain the HpaII, BGT, and MspI libraries. However, the BS-Seq data still has to be transformed into BED-format.

Many of the methylation data files are very large, especially if the datasets are generated from whole genome sequencing. Therefore, there is a big advantage for users to input compressed data. Hence, the option *-z* followed by *yes* can be used to specify that the input methylation data files are in gzip format. The arguments *-g*, *-hu*, and *-n* are the same in this task as they are in the **Gene Annotation** task. Whether the methylation data samples come from human or mouse, and whether the chromosome names are numerical or not, are used for the same reasons as in the previously described task as well. This includes the naming of the chromosomes in the output files and the difference of the number of chromosomes between human and mouse samples. If the HMST-Seq input data is not normalized, then users can apply quantile normalization on the three libraries before calculating 5mC and 5hmC levels, with the optional argument *-qn* followed by *yes*. In the next task, which is the task identifying overlaps between the genomic regions from **Gene Annotation** and 5mC/5hmC sites, the intersect feature of BEDTools can be influenced by extending the methylated sites such that sites located just outside of genomic regions will have a higher probability of being included in a cluster, or MR. The argument *-e* can be used to specify the extension of sites. This extends the methylated sites *e* bps in each direction.

The output files of this task are the 5mC and 5hmC BED-formatted files, for each of the input methylation samples. If the input methylation files come from BS-Seq, then 5hmC data will not be exported, because the bisulfite sequenicing technique does not differentiate 5hmC and 5mC. As previously mentioned, we try to make the user interaction of the whole pipeline as simple as possible. Hence, two list files containing the file

Table 6.2: The **Data Preprocessing** task input arguments. The first seven arguments are required. The rest has default values.

Default	Option	Description
None	*-fko/--fileKO*	KO condition methylation file
	-fwt/--fileWT	WT condition methylation file
	-m/--methylated	*yes* if methylation files already contains methylation-levels, *no* if not
	-z/--gzip	*yes* if the methylation files are in gzip format, *no* if they are not
	-g/--genomeFile	chromosome sizes file
	-hu/--human	*yes* if sample is from human, *no* for mouse
	-n/--numericalChr	*yes* if the chromosome names of the chromosome sizes file is numerical, *no* if they are not
no	*-qn/--quantileNorm*	*yes* if quantile normalization should be run on methylation data, *no* if not
0	*-e/--bpExtension*	bp extension from 5mC/5hmC site
1	*-c/--cutoff*	above this level: significantly modified sites
out	*-F/--folderOut*	the name of the output folder

names of the 5mC and 5hmC outputs are created: one for KO condition samples and one for WT condition samples. As seen in Figure 6.1, this task also export count files, which are files that contain the number of all 5mC/5hmC sites and significantly modified 5mC/5hmC sites in each chromosome. In addition, they include chromosome sizes, which are extracted from the chromosome sizes file. Users can define the level of significantly modified sites with the argument *-c*. The count files are used when creating the figures in the **Plotting** task. The file names of these files are also listed in a list file.

6.3 Find MRs

As described in section 5.4.1, there are multiple steps of the approach to locate MRs. In Table 6.3, all input arguments of the **Find MRs** task are shown. Users can input the KO and WT condition samples' 5mC/5hmC data files, which are the output from the **Data Preprocessing** task, with the arguments *-fko* and *-fwt* followed by the respective list file. The genomic region files, which are the output from the **Gene Annotation** task, are input to this task with the argument *-reg*. When Identifying 5'-UTR regions when doing gene annotation, we do not remove genes overlapping with 5'-UTR regions. Therefore, before finding MRs in 5'-UTR regions, we discard other genes from these regions. We do this by finding overlaps between all 5'-UTR regions and the reference file, only keeping the regions having no overlap with genes in the reference. We use BEDTools intersect including

the option *-v* to do this. Hence, the BED-formatted reference file, created by the **Gene Annotation** task, is needed for this task and must, therefore, be input with the argument *-ref*. Users can input a tissue-specific enhancer file with the argument *-e*. Because enhancer regions can not be located based on the reference genome, users must provide these files themselves.

When mapping 5mC/5hmC sites to the genomic regions using BED-Tools intersect, one optional input argument is *-f*, which can be explained as the minimum overlap required as a fraction of genomic regions. As long as the 5mC/5hmC sites are single sites and are not extended as described in the previous section: the **Data Preprocessing** task, this number will not make a difference in the outcome. However, if the 5mC/5hmC sites are extended in each direction, and we increase the *-f* argument, there must be more than just 1 bp overlap for a region to be included in the output. For example, if this input is set to 0.5, this means that at least 50% of an extended methylated site must overlap with the genomic region.

The **Find MRs** task can be quite computationally heavy. Therefore, there is an option for parallelizing the work. By setting the argument *-p* to be bigger than 1, users decide how many processes will run in parallel. If there are five genomic region files, and two samples such as one KO condition sample and one WT condition sample, both samples having 5mC and 5hmC sites, there will be $5 * 2 * 2 = 20$ combinations where MRs need to be found. The work is then split between the processes by each process finding MRs in one combo at a time. When a process is finished, it starts finding MRs in the next combination. To parallelize this task, we have used Pythons multiprocessing module, which spawns subprocesses. Subprocesses are spawned by the Pool object of the module, while the file combinations are distributed among the processes by the module's map function. Some of the genomic regions, such as 5'-UTR, usually cover a larger area of a genome. Thus, it often contains a higher amount of methylation clusters and MRs, which makes the computations more time demanding. Therefore, we sort the input data by size, largest to smallest, before beginning the MR finding. By doing this, when we have multiple processes, the most computationally heavy jobs will start first. Optimally, this makes the most computationally heavy jobs run while the rest of the processes work on the less computationally heavy jobs, as illustrated in Figure 6.3. In this figure, we see that by starting the most substantial jobs right away, as in in block **1**, all jobs are finished earlier than if the jobs were distributed randomly to the processes, as in block **2**. This procedure, i.e., sorting the files by size, is implemented for all tasks in the pipeline that have the option of parallelizing. In chapter 7, we will have a look at whether this works as intended or not.

MRs are, as we know, defined by a minimum number of consecutive 5mC/5hmC sites having a maximum bp inter-distance, within a genomic region. The minimum number of consecutive sites of each MR can be set with the arguments *-mc1* for TSS and TES regions, *-mc2* for gene body, 5'-UTR, and intergenic regions, and *-mc3* for enhancers. We have grouped them this way because of the length of the various regions. TSS and TES regions are usually shorter than gene body, 5'-UTR, and intergenic regions.

Table 6.3: The **Find MRs** task input arguments.

Default	Option	Description
None	*-fko/--koFiles*	the list file from the **Data Preprocessing** task containing 5mC/5hmC file names for KO condition samples
	-fwt/--wtFiles	the list file from the **Data Preprocessing** task containing 5mC/5hmC file names for WT condition samples
	-reg/--regionFiles	the list file from the **Gene Annotation** task containing genomic region file names
	-ref/--reference	the filtered and BED-formatted reference file
	-e/--enhancerFile	BED-formatted enhancer region file
1e-9 (i.e. 1bp)	*-f/--minOverlap*	minimum overlap required as a fraction of the region file, for BEDTools intersect when finding overlaps between methylated sites and regions
1	*-p/--numProcesses*	the number of processes running this task at the same time
3	*-mc1/--minCons1*	the minimum number of consecutive methylated sites in TSS and TES regions to be considered MRs
5	*-mc2/--minCons2*	the minimum number of consecutive methylated sites in gene body, 5'-UTR and intergenic regions to be considered MRs
3	*-mc3/--minCons3*	the minimum number of consecutive methylated sites in enhancer regions to be considered MRs
2000	*-a/--adjacency*	the maximum number of bps between adjacent sites in an MR
out	*-F/--folderOut*	the name of the output folder

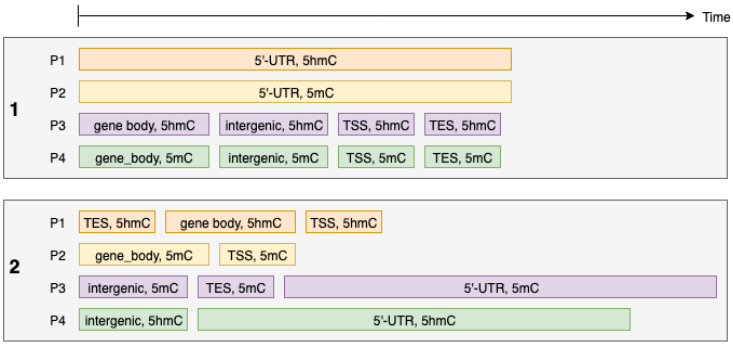

Figure 6.3: When having multiple processes, we can sort the data by size before distributing the jobs to the processes, such that the biggest jobs will start first. Here we see the hypothetical difference of sorting the data by size (**1**) and distributing data randomly (**2**) between four processes before starting finding MRs. The genomic regions and methylation states for the different job-sizes used in this figure are purely chosen to illustrate the idea.

The argument *-a* is used to set the maximum inter-distance between sites of MRs.

The output data of this task are two sets of data: one set containing MRs for KO samples, and one set containing MRs for WT samples, as well as the respective list files.

6.4 Preparation for DMR Search

As we know, DMR finding is a process of comparing MRs between the two different biological condition/samples. The input arguments *-ko* and *-wt* are used for inputting the two list files containing file names for MRs in KO and WT samples, respectively. These two, as well as the rest of the input arguments, are shown in Table 6.4. By default, missing values are replaced by zeros. However, another imputation method can be used, such as the median method. It replaces missing values by the median of the 5mC/5hmC levels of an MR. Whether to use zeros or median imputation is specified by the optional argument *-i* followed by either *zeros* or *median*. The arguments *-p*, *-mc1*, *-mc2*, *-mc3*, *-a*, and *-F* are the same as described in the **Find MRs** task. We need these values here as well, for the reasons described under the Missing Data section of section 5.4.2.

The main output data of this task are the overlapping MRs and a list file. Additionally, a dataset containing the necessary information for plotting the distribution of 5mC/5hmC levels of overlapping MRs is output, but only if there is no more than two samples present. If there are more than one KO or WT sample, this output will not be created. If there is just one of each condition sample, a list file containing the file names of the dataset will also be created.

Table 6.4: Input arguments for the **Preparation for DMR Search** task.

Default	Option	Description
None	*-ko/--KOfilesMRs*	the list file from the **Find MRs** task containing file names for MRs of KO condition samples
	-wt/--WTfilesMRs	the list file from the **Find MRs** task containing file names for MRs of WT condition samples
zeros	*-i/--impute*	what to impute missing values by
1	*-p/--numProcesses*	the number of processes
3	*-mc1/--minCons1*	the minimum number of consecutive methylated sites in TSS and TES regions to be considered MRs
5	*-mc2/--minCons2*	the minimum number of consecutive methylated sites in gene body, 5'-UTR and intergenic regions to be considered MRs
3	*-mc3/--minCons3*	the minimum number of consecutive methylated sites in enhancer regions to be considered MRs
2000	*-a/--adjacency*	the maximum number of bps between adjacent sites in an MR
out	*-F/--folderOut*	the name of the output folder

6.5 DMR Search

All input arguments of this task are shown in Table 6.5. The overlapping MRs created in the previous task are input to this task with the argument *-f* followed by the list file containing the overlapping MRs file names. To set the significance level for the hypothesis tests checking for DMRs, the optional argument *-cp* can be used. All results having *p*-value lower than the significance level are considered as significant, and the null hypothesis, that the 5mC/5hmC levels of the samples are similar, is rejected. There is also an option for parallelizing this task, with the optional argument *-p*. For calculating the *p*-value for overlapping MRs, i.e., finding DMRs, a user can choose between several statistical test methods. The three main test methods in the pipeline are: Pranksum, Mranksum, and Rranksum. The default is Pranksum, where the computation method of the *p*-value is as following:

$$p = \begin{cases} exact & \text{if} \quad n_x < 10 \\ mannwhitneyu & \text{otherwise} \end{cases}$$

where n_x is the number of 5mC/5hmC sites in the KO sample of the overlapping MR, *exact* is our implementation of the exact enumeration method described in section 2.3.3, and *mannwhitneyu* is the mannwhitneyu function of the Python package scipy.stats. This function performs a two-sided Mann-Whitney U test. Because we impute missing values in overlapping MRs, the samples sizes in an overlapping MR are the same. Therefore, n_x and n_y are the same, and it does not matter which we use when choosing computation method. The scipy.stats.mannwhitneyu method is only advised to use when the number of observations in each sample is > 20. However, we have based the *Pranksum* approach on the MATLAB function ranksum which, for us, has the same computation method as our *Pranksum*. In the pipeline, using MATLAB's ranksum (Mranksum) is the next test method option, and the computation method is as following:

$$p = \begin{cases} exact & \text{if} \quad min(n_x, n_y) < 10 \quad \text{and} \quad n_x + n_y < 20 \\ approximate & \text{otherwise} \end{cases}$$

where *exact* is the exact computation of the *p*-value and *approximate* is the two-sided Wilcoxon rank-sum test. Mranksum and Pranksum are similar in the way *p*-values are calculated for different sample lengths. Because MATLAB is a proprietary programming language, not all users might have access to it. Therefore, we have a third test method, Rranksum, which is the wilcox.test function of R. This is also a two-sided Wilcoxon rank-sum test, which is utilized by calling Rscript. Rscript is an alternative front end for R. The default computation method of the *p*-value of Rranksum is as following:

$$p = \begin{cases} exact & \text{if} \quad n_x < 50 \quad \text{and} \quad n_y < 50 \quad \text{and} \quad \text{no ties} \\ approximate & \text{otherwise} \end{cases}$$

where *exact* and *approximate* are the same as in the other two test methods. However, in this method, an exact computation of the *p*-value is used if the sample size is less than 50. This means the Rranksum could potentially be very slow compared to Pranksum and Mranksum. Also, the since the computation methods are different between the three test methods, the results might not be the same. In the next chapter, we will do a test run on some real data, and show the results and time consumption for the three test methods.

All approximation methods used by the three test methods, Pranksum, Mranksum, and Rranksum, use a normal approximation for calculating the *p*-value. The methods also apply a continuity correction. This correction allows for the fact that the normal distribution is continuous, whereas the distribution of the test statistic is discrete [41].

Users can check the quality of DMRs, by using the optional argument, *-pl* followed by *yes*. This plots all DMRs, and an example of such a plot is shown in Figure 6.4. Here, we see that the *p*-value of this DMR is 0.0285. In the bottom right we see that the WT sample of this DMR was missing a value, which has been imputed by a zero. By showing these plots, users can check the significance of DMRs, and may run the task again by fine-tuning the input arguments, such as changing *-cp* adding a false discovery rate control, or change the imputing method for missing values. However, if there are many DMRs, plotting all of them might take a while.

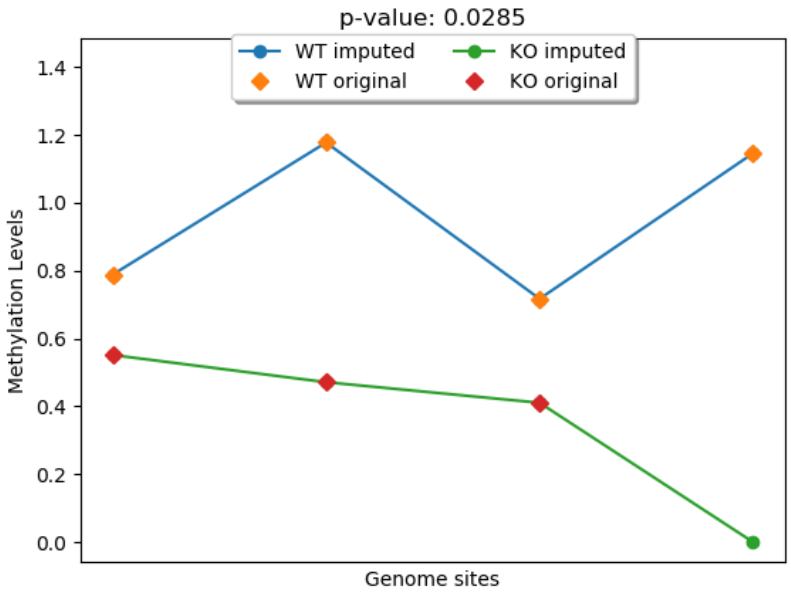

Figure 6.4: One DMR containing two samples' 5mC levels.

When all DMRs are found, the results are exported to multiple files, where each combination of KO and WT sample, genomic region, and methylation state has their own dataset consisting of:

Table 6.5: Input arguments for the **DMR Search** task.

Default	Option	Description
None	*-f/--prepfile*	the list file from the **Preparation for DMR Search** task containing file names for overlapping MRs
0.05	*-cp/--cutoffPval*	cut off significance level
1	*-p/--numProcesses*	the number of processes
Pranksum	*-T/--testMethod*	which statistical hypothesis test method to run
None	*-fdr*	the false discovery rate for the Benjamini/Hochberg method
no	*-pl/--plot*	*yes* if figures of each DMR should be created, *no* if not
out	*-F/--folderOut*	the name of the output folder

- A CSV file containing all DMRs
- A CSV file containing all hyper DMRs
- A CSV file containing all hypo DMRs
- A CSV file containing all gene names of hyper DMRs
- A CSV file containing all gene names of hypo DMRs

6.6 Plot Preparation

The **Plot Preparation** task has two main functions: create files containing MR counts and convert the format for MRs in TSS, gene body, TES, and enhancer datasets. In Table 6.6, the input arguments of this task are shown. The two list files containing MRs of KO and WT condition samples, as created in the **Find MRs** task, are input by the arguments *-ko* and *-wt*, respectively. As in the **Data Preprocessing** task, the cut off for significantly modified sites can be specified by the input argument *-c*.

Table 6.6: Input arguments for the **Plot Preparation** task.

Default	Option	Description
None	*-ko/--KOfilesMRs*	the list file from **Find MRs** task containing file names for MRs of KO condition samples
	-wt/--WTfilesMRs	the list file from **Find MRs** task containing file names for MRs of WT condition samples
1	*-c/--cutoff*	above this level: significantly modified sites
out	*-F/--folderOut*	the name of the output folder

The output MR count files contain information of the total number of methylated sites in MRs as well as the number of significantly modified sites in MRs for each genomic region and sample. These count file names are stored in a list file. In this task, the data of MRs from the **Find MRs** task is converted to only contain the necessary information for showing the distribution of 5mC/5hmC sites, such as one row per MR. In this way, it reduces the computational load for the **Plotting** task. All file names of these converted datasets are listed in a list file. The MR count files are also stored in a separate list file.

6.7 Plotting

The set of input arguments for the **Plotting** task is the most extensive, where users have the opportunity to input all of the count files, the list file for the converted MRs dataset from the **Plot Preparation** task, and the list file for the overlapping MRs dataset from the **Preparation for DMR Search** task. All input arguments for this task is shown in Table 6.7. The first four arguments are the various count files, while the fifth and sixth arguments are the list file of the converted MRs dataset and the overlapping MRs dataset, respectively. Users can input all files at the same time, which will create all possible figures, but the **Plotting** task can also be run with one or more input files. Not inputting all files will result in just the respective figures being created. The input arguments used for creating the various figures, are:

- **The distribution of significantly modified 5mC/5hmC MRs**: *-reg* and *-sit*
- **The DMR and DhMR distribution**: *-cmc* and *-chmc*
- **The distribution of 5mC and 5hmC levels in MRs of TSS, gene body, TES and enhancer region**: *-aMR* and/or *-oMR*. The *-aMR* argument is used for inputting the list file containing prepared MRs, which is output from the **Plot Preparation** task. The *-oMR* can only be used if there are two samples: one KO and one WT. The difference between the two is that the *-aMR* argument file should contain 5mC/5hmC sites of all MRs, while the *-oMR* argument file should contain 5mC/5hmC sites of overlapping MRs. The two arguments *--plotENH* and *--plotTGT* can be used to only create either the TSS, gene body, and TES plots, or the enhancer plots.

  The two input arguments *-gX* and *-gY* shall have the same values as that of the *-X* and *-Y* arguments in **Gene Annotation** task. These are used to map 5mC/5hmC sites of TSS and TES MRs to the same region span, as discussed in section 5.5.3. Since the gene body regions may differ in length, we have to map them to the same span as well. The size of this span can be set by the user with the argument *-G*. Another two arguments, *enhX* and *enhY*, are used to map enhancer MRs to the same region span. When plotting the distribution of

5mC/5hmC levels of MRs in enhancers, it is possible to only show tissue-specific MRs by using the *-t* option. Only the 5mC/5hmC levels of the selected tissues will then be shown, such as cerebellum, liver, or lung.

When we increase the number of methylated sites in each MR to make it denser, we use a one-dimensional nearest neighbor interpolation algorithm. For influencing the number of estimated data points, users can set the step size with the optional argument *-w*. The lower the step size, the higher the density of methylation levels. When the data has been increased, and the average of all methylation levels at each site has been taken, we use a centered moving average for smoothing the averages. By changing the input argument *-ws*, users can the subset size as described in section 2.5.1. The option *-s* is the sigma, i.e., the standard deviation of the Gaussian, used when smoothing the data further using the one-dimensional Gaussian filter.

As mentioned earlier, outputs of this section are four different types of figures, which show: (1) the distribution of DMRs and DhMRs, (2) the density of significantly modified sites in MRs, (3) the distribution of 5mC/5hmC levels in MRs of TSS, gene body, and TES, and (4) the distribution of 5mC/5hmC levels in enhancer MRs. All results generated in this task are also exported to text files, which allows users to investigate the results further, such as using other Python scripts, MATLAB scripts, or any other tools to reproduce the figures.

Table 6.7: Input arguments for the **Plotting** task.

Default	Option	Description
None	*-reg/--listCountMRs*	list file from the **Plot Preparation** task containing MR counts
	-sit/--listCountSites	list file from the **Data Preprocessing** task containing 5mC/5hmC site counts
	-cmc/--DMRmC	count file from the **DMR Search** task containing DMR counts
	-chmc/--DMRhmC	count file from the **DMR search** task containing DhMR counts
	-aMR/--allMRs	list file from the **Plot Preparation** task containing MRs of TSS, gene body, TES and enhancers
	-oMR/--overlappingMRs	list file from the **Preparation for DMR Search** task containing overlapping MRs
yes	*--plotTGT*	*yes* if TSS, gene body, and TES should be plotted, *no* otherwise
1000	*-gX*	the number of upstream bp to the transcription start/end site
1000	*-gY*	the number of downstream bp from the transcription start/end site
4000	*-G*	the size of the span gene body will be mapped to
yes	*--plotENH*	*yes* if enhancer should be plotted, *no* otherwise
5000	*-enhX*	the number of upstream bp to the center of enhancer regions
5000	*-enhY*	the number of downstream bp from the center of enhancer regions
all	*-t/--tissue*	enhancers: what tissue to plot
100	*-w*	step size when interpolating TSS, gene body, TES, and enhancer MRs
500	*-ws*	window size for centered moving average smoothing
50	*-s/--sigma*	standard deviation for Gaussian kernel, for smoothing TSS, gene body, TES, enhancers
out	*-F/--folderOut*	the name of the output folder

Part III

Results and Conclusion

Chapter 7

Results and Discussion

In this chapter, we will demonstrate how to set up an Abel environment to run the pipeline, show the test runs of the pipeline where user interaction is demonstrated, present results, and show the result figures. Additionally, we will look into the execution times and memory consumption for a few computationally heavy tasks in the pipeline, and compare the results of the three statistical test methods for differential methylation analysis. In the end, results of HMST-Seq-Analyzer is compared to that of the popular methylation analysis package - methylKit by using WGBS data.

7.1 Setting Up Pipeline Environment

A Python 2.7 running environment is installed by the conda package manager through the Miniconda installer, where the new pipeline will be tested. Many additional Python-packages are included when installing conda. However, some of the packages (e.g., SciPy and BEDTools) have to be installed by conda as following:

```
$ conda install scipy
...
$ conda install -c bioconda bedtools
...
```

When installing BEDTools through conda, the "-c bioconda" means that we want to install BEDTools through the channel named Bioconda. Bioconda is a channel for the conda package manager specializing in bioinformatics software [42].

UiO has a site license for MATLAB, which means it is already installed in Abel. However, Python's MATLAB engine has to be installed in the local directory. First, the user has to load the MATLAB module file with the following command:

```
$ module load matlab
```

To install the MATLAB engine, we have to go to the directory "matlabroot/extern/engines/python", where matlabroot is the MATLAB root directory, which in our case is "/cluster/software/VERSIONS/matlab/R2017a/", and enter the following command:

```
$ python setup.py build --build-base="$HOME" install
```

Since users do not have the write permission in the MATLAB folder, the engine has to be installed in the users' home directory ($HOME). R is a free software under the GNU General Public License of Free Software Foundation and is already available in Abel. The various packages, modules, and software that is needed by HMST-Seq-Analyzer, and their versions are shown in Table 7.1.

Table 7.1: Software versions used for pipeline environment in Abel.

Software	**Version**
conda	4.6.14
Python	2.7.15
argparse	1.1
multiprocessing	0.70a1
pandas	0.23.4
NumPy	1.15.4
Matplotlib	2.2.3
seaborn	0.9.0
SciPy	1.2.1
MATLAB	R2017a
BEDTools	2.27.1
R	3.5.1

After all software and packages were installed, we navigated to the pipeline package folder and installed the pipeline, as described in section 4.1.

7.2 Test Data

The first test run of the new pipeline is based on our in-house mouse HMST-Seq data. In this thesis, we have masked the real sample names and conditions, and we only show results from chromosome 1, because it is confidential. In section 7.10, a full run on public human ENCODE data will be illustrated.

The mouse dataset consist of one KO condition sample and one WT condition sample, where both samples have already been normalized across the three tag count-libraries by the GRSN algorithm [14], as described in section 2.2.1. The file names are KO.chr1.txt and WT.chr1.txt for the KO condition sample and WT condition sample, respectively. The directory tree for the files of this test run is shown in Figure 7.1. In the mouse folder, the file mm10.enhancers.bed is the BED-formatted tissue-specific enhancers file, mm10.sizes_numerical is the sorted mm10 mouse assembly chromosome sizes file, while mm10.refFlat.txt is the cleaned mm10 reference genome file.

```
in_data
`- mouse
    |-- mm10.enhancers.bed
    |-- mm10.sizes_numerical
    |-- mm10.refFlat.txt
    `-- chr1
    |   |-- KO.chr1.txt
    |   `-- WT.chr1.txt
```

Figure 7.1: The input data directory tree for the mouse dataset.

7.3 Test Run

To demonstrate the analysis pipeline run in Abel, a batch script for analyzing the aforementioned mouse dataset is shown here:

```
#!/bin/csh

#Job name:
#SBATCH --job-name=HMST-mouse-chr1
#
# Project:
#SBATCH --account=UIO
#
# Wall clock limit:
#SBATCH --time=10:00:00
#
# Max memory usage:
#SBATCH --mem-per-cpu=4G
#
# Number of cores:
#SBATCH --cpus-per-task=4
#
# Number of nodes:
#SBATCH --nodes=1

# Do the work
/usr/bin/time -p hmst_seq_analyzer gene_annotation\
-F mouse_chr1 -hu no -n yes\
-r in_data/mouse/mm10.refFlat.txt\
-g in_data/mouse/mm10.sizes_numerical
echo gene_annotation-DONE

/usr/bin/time -p hmst_seq_analyzer data_preprocessing\
-F mouse_chr1 -z no -m no -hu no -n yes\
-fko in_data/mouse/chr1/KO.chr1.txt\
-fwt in_data/mouse/chr1/WT.chr1.txt\
-g in_data/mouse/mm10.sizes_numerical
echo data_preprocessing-DONE

/usr/bin/time -p hmst_seq_analyzer find_MRs\
-F mouse_chr1 -p 4\
-fko mouse_chr1/list_mC_hmC_files_KO.txt\
-fwt mouse_chr1/list_mC_hmC_files_WT.txt\
-ref mouse_chr1/data/mm10.refFlat_clean_sorted.bed\
-reg mouse_chr1/list_region_files.txt\
-e in_data/mouse/mm10.enhancers.bed
echo find_MRs-DONE
```

```
/usr/bin/time -p hmst_seq_analyzer prepare_for_DMR_finding\
-F mouse_chr1 -p 4\
-ko mouse_chr1/list_all_filtered_formatted_MRs_KO.txt\
-wt mouse_chr1/list_all_filtered_formatted_MRs_WT.txt
echo prepare_for_DMR_finding-DONE

/usr/bin/time -p hmst_seq_analyzer DMR_search\
-F mouse_chr1 -p 4\
-f mouse_chr1/list_prepared_for_DMR_finding_imputed.txt
echo DMR_search-DONE

/usr/bin/time -p hmst_seq_analyzer prep4plot\
-F mouse_chr1\
-ko mouse_chr1/list_all_filtered_formatted_MRs_KO.txt\
-wt mouse_chr1/list_all_filtered_formatted_MRs_WT.txt
echo prep4plot-DONE

/usr/bin/time -p hmst_seq_analyzer plot_all\
-F mouse_chr1\
-reg mouse_chr1/list_count_allMRs_regions_files.txt\
-sit mouse_chr1/list_count_sites_files.txt\
-cmc mouse_chr1/counts_DMR_hypo_hyper_imputed_Pranksum_5mC.csv\
-chmc mouse_chr1/counts_DMR_hypo_hyper_imputed_Pranksum_5hmC.csv\
-aMR mouse_chr1/list_TSS_genebody_TES_enhancer_allMRs.txt\
-oMR mouse_chr1/list_overlapping_MRs.txt
echo plot_all-DONE
```

Here, we allocated four CPUs on one node with 4GB memory per CPU. The three tasks: **Find MRs**, **Preparation for DMR Search**, and **DMR Search** are run with four processes each. The name of the output folder is mouse_chr1, as specified by the *-F* argument in the tasks. We added */usr/bin/time -p* at each task so that the execution times are included in the standard output which is directed to a "slurm-%j.out" file, where "%j" is replaced with the job allocation number. By adding "echo %t-DONE", where "%t" is the task name, it allows users to track the progress of the pipeline run. Here, the default values are used, as described in Chapter 6, except for arguments such as output file name and the number of processes. The text files having names starting with "list_," are the list files containing file names of the respective datasets.

7.4 Result Figures

After running the full pipeline based on the above-mentioned description, eight figures were created in the folder mouse_chr1/plots/. Here, we present each of the figures and evaluate the significance of the results. In the next sections of this chapter, several input arguments of the pipeline will be changed in order to compare the difference in the results.

7.4.1 DMR and DhMR Distribution

Figure 7.2 shows the distribution of DMRs and DhMRs in different genomic regions, where hypo and hyper DMRs/DhMRs are separated. This figure

suggests that TSS regions have the highest percentages of hyper DMRs, where ~4.6% of the MRs are differentially methylated. The percentage of hypo DhMRs is also high in the TSS regions: ~3%.

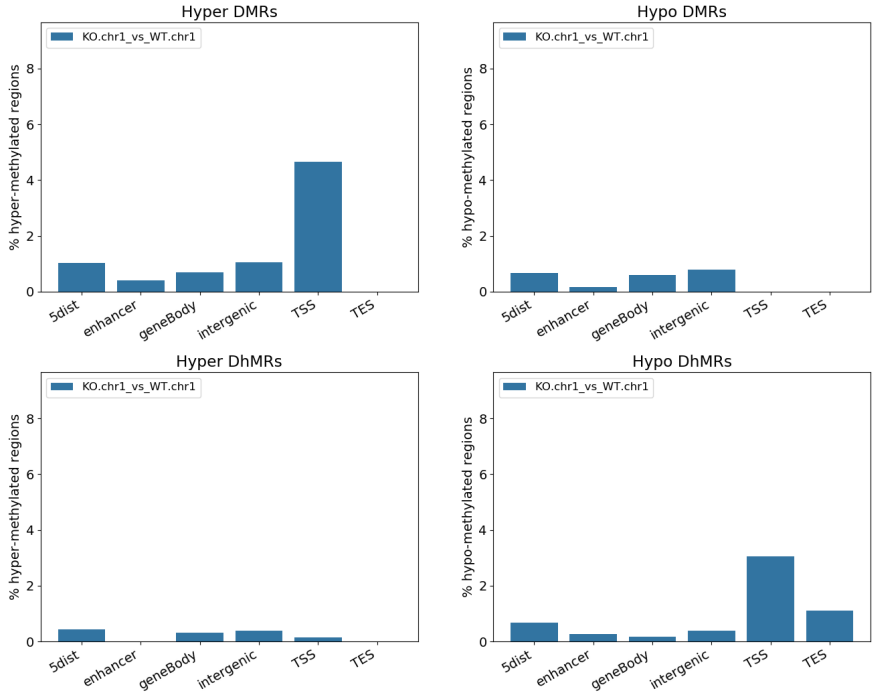

Figure 7.2: The distribution of hypo and hyper DMRs and DhMRs within different genomic regions. 5dist on the x-axes is the same as 5'-UTR.

7.4.2 Relative Density for MRs

Figure 7.3 is the second figure, which shows the distribution of significantly modified 5mC and 5hmC MRs among different genomic regions, as described in section 5.5.2. Approximately 50% of the raw 5mC and 5hmC sites are significantly modified for both samples, as can be seen in the bars named "genome". The bars of 5'-UTR, TSS, gene body, TES, intergenic, and enhancer are for 5mC and 5hmC sites located in MRs.

From the figure we see that the number of significantly modified 5mC and 5hmC sites is distributed equally in genome, 5'-UTR, gene body, TES, intergenic, and enhancer regions between WT and KO sample conditions. However, in the TSS regions, the dynamical changes of 5mC or 5hmC often occur. Especially, there is anti-correlation between the 5mC and 5hmC levels in TSS regions. If 5mC is higher in KO sample condition than that in WT sample condition, then you will expect 5hmC to be lower in KO than in the WT condition. This result is expected because TSS regions usually control the downstream gene regulation. Especially, for DNA methylation in mammals, it is well known that 5mC and 5hmC are important in controlling gene regulation [43].

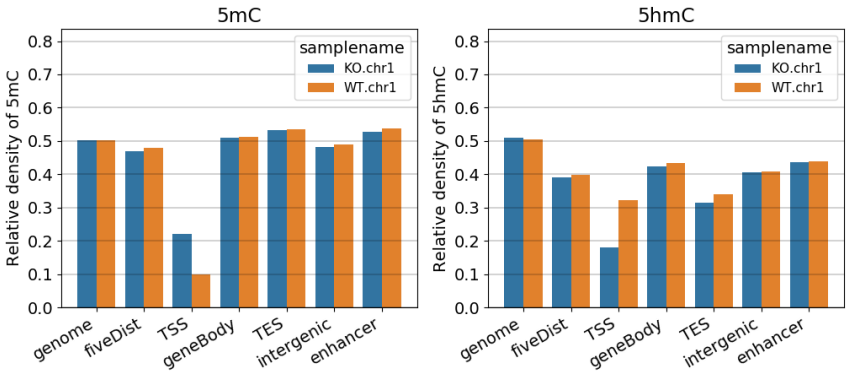

Figure 7.3: The relative density of significantly modified sites in MRs within genomic regions. The bars named "genome" are for all methylated sites, not just sites that are in MRs.

7.4.3 Distribution of 5mC and 5hmC in TSS, Gene Body, and TES regions

These figures come in pairs, one for 5mC and one for 5hmC, as described in section 5.5.3. The distribution of 5mC and 5hmC are shown in Figure 7.4 and Figure 7.5, respectively. In both figures, there is a relatively high correlation between KO and WT samples, except for in the TSS regions. In the TSS regions, there is a higher 5mC level in the KO condition sample than that in the WT condition sample. It indicates that there is an increase in 5mC but a decrease of 5hmC in TSS regions after the gene knockout. These results match with the distribution of significantly modified 5mC and 5hmC in TSS regions in Figure 7.3. Here, figures generated by only considering overlapping MRs are not shown because they are almost the same as the figures based on all MRs.

7.4.4 Distribution of 5mC and 5hmC MRs in enhancer regions

The average 5mC and 5hmC levels in enhancer regions are the last two figures created by the analysis pipeline, which are shown in Figure 7.6 and Figure 7.7, respectively. Often, there is a low 5mC/5hmC in the center of enhancer regions, but since we only use chromosome 1 data, this might not show here. In both figures, there is a high correlation between the samples.

7.5 Performance Comparison

Even though the HMST-Seq method sequences cytosines, methylated cytosines, and hydroxymethylated cytosines at CG sites only, the data size can still be large when doing whole genome-wide sequencing. In order to reduce the heavy computation, parallel computation can be applied on three of the pipeline tasks (**Find MRs**, **Preparation for DMR Search**, and **DMR Search**) by adjusting input argument *-p*, as described in Chapter 6. By using output results from the tasks **Gene Annotation** and **Data**

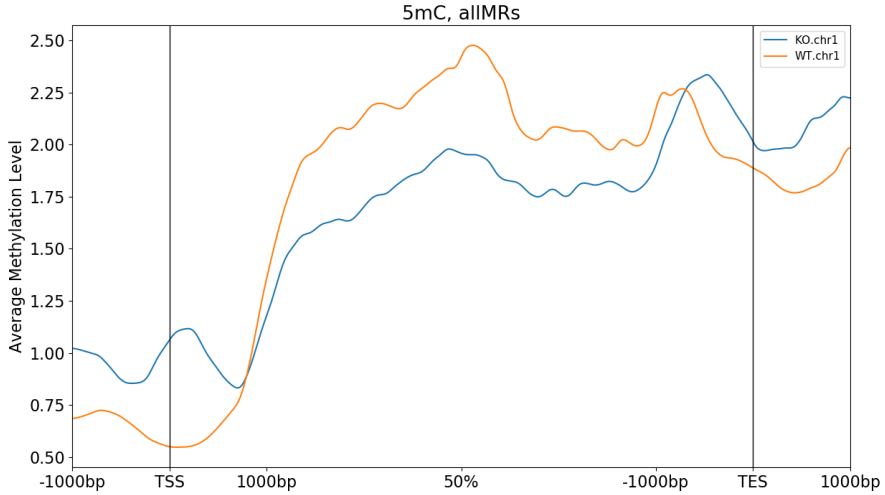

Figure 7.4: The average 5mC levels of MRs in the combined TSS, gene body, and TES regions.

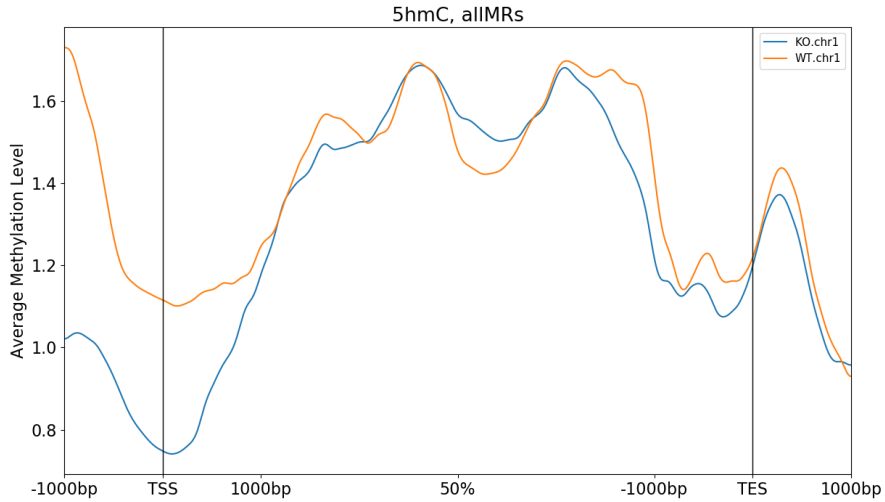

Figure 7.5: The average 5hmC levels of MRs in the combined TSS, gene body, and TES regions.

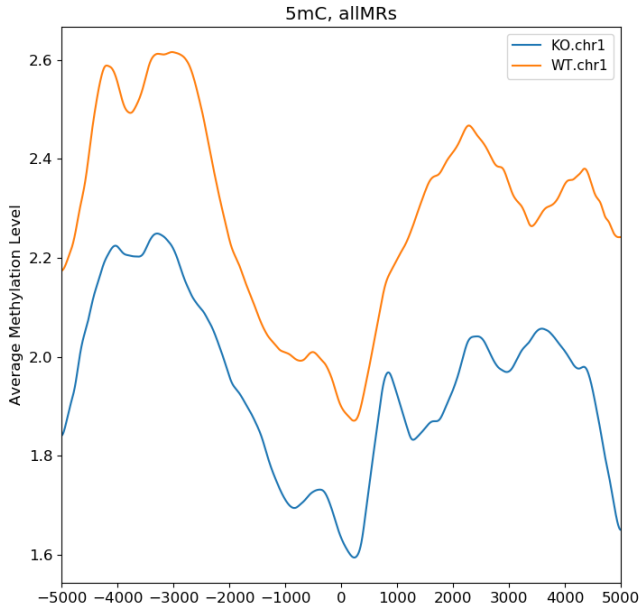

Figure 7.6: The average 5mC levels of MRs in enhancer regions.

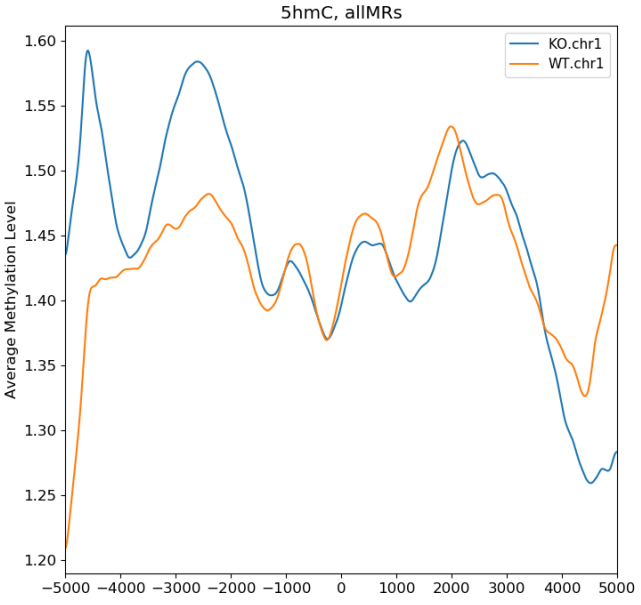

Figure 7.7: The average 5hmC levels of MRs in enhancer regions.

Preprocessing, we ran the aforementioned three tasks in parallel with default input arguments, but where the number of processes was modified from 1 to 10. We created 10 batch scripts for each of the three tasks, where we used the commands *SBATCH –mem-per-cpu=3G*, *SBATCH –nodes=1*, and *SBATCH –cpus-per-task=%j*, where we changed %j depending on the number of processes. We also changed the input argument *-p* to be the same as %j. This run was done using Pranksum. However, later, we will compare time and memory consumption between the three test methods: Pranksum, Mranksum, and Rranksum. The execution time of the different number of processes is seen in Figure 7.8, where execution time is the wall clock time from start to end of the specific pipeline task in seconds. From this figure, we see that, for all three tasks, the most significant performance increase, in terms of execution time, is when going from using one process to using two parallel processes. There is also a significant increase in performance, going from two to three parallel processes. However, for **Preparation for DMR Search** and **DMR Search** the performance does not increase when having more than three processes. For **Find MRs**, five or six processes seems to be a sufficient number. In Figure 7.9, we see the memory consumption for the same three tasks across the various number of processes. The memory consumption is fairly stable in all three tasks, where increasing the number of processes will linearly increase the memory consumption.

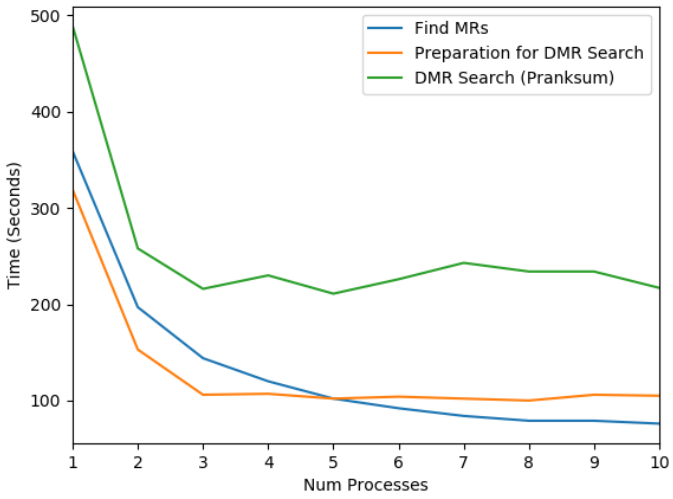

Figure 7.8: Time consumption for the three pipeline tasks **Find MRs**, **Preparation for DMR Search**, and **DMR Search** with various number of processes.

7.6 Comparison of Execution Time for Different Data Sizes

The results from the previous section suggest that it is the most computational efficient when running the pipeline on HMST-Seq data for two sam-

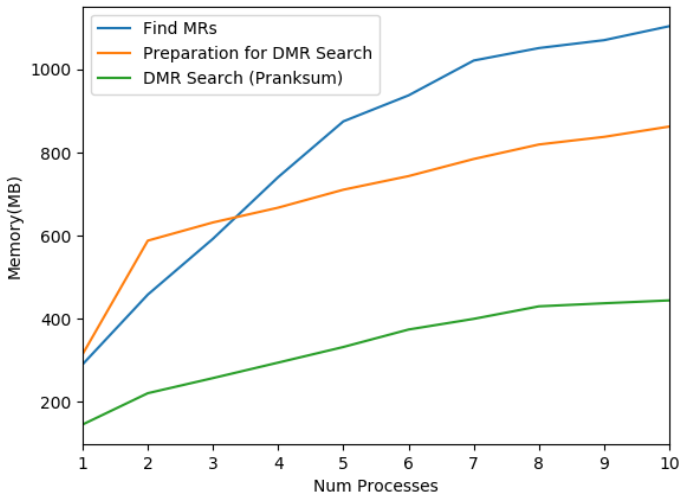

Figure 7.9: Memory consumption for the three pipeline tasks **Find MRs**, **Preparation for DMR Search**, and **DMR Search** with various number of processes.

ples with five, three, and three processes in the tasks **Find MRs**, **Preparation for DMR Search**, and **DMR Search**, respectively. To check the CPU times required for a large dataset, the full pipeline was tested twice: one for chromosome 1 only, and the other for all 25 chromosomes. For both batch scripts, we used the following instructions for the queue system:

```
# Max memory usage:
# SBATCH --mem-per-cpu=4G
#
# Number of cores:
# SBATCH --cpus-per-task=5
#
# Number of nodes:
# SBATCH --nodes=1
```

The resulting execution times of all tasks, as well as the total execution time of the two runs, are shown in Table 7.2. Here, the dataset only containing chromosome 1 tag counts from mouse genome, have sizes 4.4 MB and 4.5 MB for the KO and WT samples, respectively. For the dataset containing tag counts of all chromosomes, the KO and the WT condition sample sizes are 65 MB and 66 MB, respectively. This is shown in parentheses in the second column of the table. Above the parentheses, there are two numbers for each of the datasets. These are the number of raw genome sites. In the last column of the table, we show the execution times, which are also represented as a bar plot in Figure 7.10. The full pipeline took 170 minutes and 6 seconds to run for the dataset containing tag counts of all chromosomes, while it, for the dataset containing only chromosome 1 tag counts, took 10 minutes and 37 seconds. As these numbers show, the time consumption is substantially higher for the full dataset. If we run the full pipeline with more than one KO sample, it will take much

more time to complete it. Thus, the current pipeline is not so efficient for analyzing very large datasets. The reason for the plotting taking such a long time is because we create all possible figures showing the distribution of 5mC/5hmC in TSS, gene body, and TES MRs for both dataset: all MRs and overlapping MRs. If a user is only interested in creating figures from only one of the sets such as all MRs, the time consumption of plotting would almost be halved. For the **Gene Annotation** task, it is just a coincidence that the execution times differ for the two datasets. This task does the exact same in both runs, and could be run only once if no input arguments are changed.

Table 7.2: Time consumption of pipeline run for two different input data sizes.

Chr	# Sites	Task	Execution Time
Chr 1	KO:70201 WT:71646 (4.4MB/4.5MB)	Gene Annotation	2m23s
		Data Processing	0m6s
		Find MRs	1m13s
		Prep for DMR Search	1m31s
		DMR Search	3m2s
		Plot Preparation	0m22s
		Plotting	1m57s
		Whole Pipeline	**10m37s**
All	KO:1037346 WT:1054613 (65MB/66MB)	Gene Annotation	2m50s
		Data Processing	0m55s
		Find MRs	26m19s
		Prep for DMR Search	37m0s
		DMR Search	53m46s
		Plot Preparation	7m23s
		Plotting	41m52s
		Whole Pipeline	**170m6s**

7.7 Method Comparisons for Finding DMRs and DhMRs

As described in section 6.5, the pipeline has an option for choosing between three test methods to use for finding DMRs. In Table 7.3 we show the number of hyper and hypo DMRs and DhMRs found by the three methods for the chromosome 1 mouse dataset. The second column shows the number of overlapping MRs found in the **Preparation for DMR Search** task. The number of hyper and hypo DMRs and DhMRs are somewhat similar across the three methods. However, for some of the genomic regions there is a difference even though all test methods are using a Wilcoxon rank-sum test. Looking at the output DMRs/DhMRs of **DMR Search** for the three methods Pranksum, Mranksum, and Rranksum, we can tell that the Mranksum method and Pranksum method find all the same DMRs and DhMRs. However, 30 of the DMRs/DhMRs found by

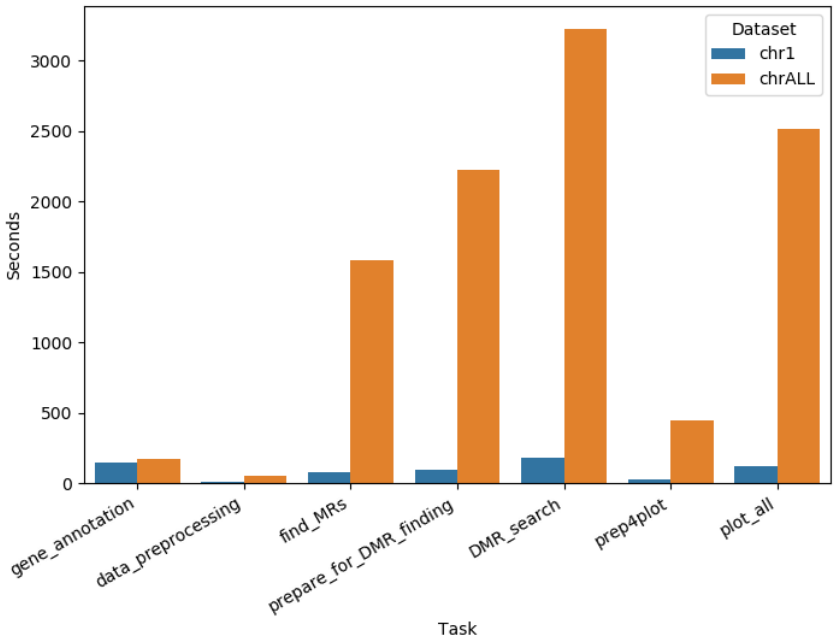

Figure 7.10: The execution times from Table 7.2, represented as a bar plot.

Mranksum and Pranksum was not in the output when using the Rranksum method. The total number of DMRs and DhMRs found by both Pranksum and Mranksum is 364 and 231, respectively, while Rranksum found 339 DMRs and 226 DhMRs. All 565 DMRs and DhMRs found by Rranksum are included in the 595 DMRs and DhMRs found by Mranksum and Pranksum. Since there is a difference in three test methods, as shown in section 6.5, the results of Rranksum often differ from that of Mranksum/Pranksum. Rranksum computes the exact p-value if the size of the samples is less than 50, and there are no ties, and uses a normal approximation otherwise. Mranksum and Pranksum compute the exact p-value if the size of the samples is less than 10, and uses a normal approximation otherwise. Because of these differences, the p-values might not be the same across the three methods for all MR comparisons. For example, if we have an overlapping MR where the sample size is six, but there is a tie, the Rranksum method will use a normal approximation for computing the p-value while the Mranksum and Pranksum methods will calculate the exact p-value.

When choosing a test method to be used in the pipeline, we should consider the time and memory consumptions as well. In Table 7.4, we have compared the execution times of running the **DMR Search** task using the dataset containing only chromosome 1 with the three test methods: Mranksum, Pranksum, and Rranksum. The instructions given to the queue system was: *–mem-per-cpu=4G*, and *–nodes=1*, but using only one cpu per task: *–cpus-per-task=1*. In the table, we see that the Rranksum method uses a lot more time than Pranksum and Mranksum. In addition, we ran the **DMR Search** task with various number of processes, using the same slurm

Table 7.3: The number of hyper and hypo DMRs and DhMRs found using the different test methods: Mranksum, Pranksum, and Rranksum, in the chromosome 1 mouse dataset.

Genomic Region	# MRs	State	# D(h)MRs					
			Mranksum		Pranksum		Rranksum	
			Hyper	Hypo	Hyper	Hypo	Hyper	Hypo
TSS	687	**5mC**	32	0	32	0	32	0
		5hmC	1	21	1	21	1	21
TES	179	**5mC**	0	0	0	0	0	0
		5hmC	0	2	0	2	0	2
Gene Body	1862	**5mC**	13	11	13	11	13	11
		5hmC	6	3	6	3	5	3
Intergenic	1519	**5mC**	16	12	16	12	14	12
		5hmC	6	6	6	6	6	5
5′-UTR	16536	**5mC**	168	112	168	112	145	112
		5hmC	73	113	73	113	73	110
Total	20783	**-**	595		595		565	

instructions as in section 7.5, and created two plots showing the time and memory consumption for the three methods. The time consumption is shown in Figure 7.11, and the memory consumption is shown in Figure 7.12. As seen in both Table 7.4 and Figure 7.11, the Rranksum method uses a lot more time than the other two methods. This may be caused by the method difference, where Rranksum uses an exact p-value calculation when the sample size smaller than 50 and there are no ties in the data, but the other two methods only use exact computation when sample size smaller than 10. As discussed in section 2.3.3, when the sample size is big, this can be very computationally heavy. The time consumption of the Mranksum method and the Pranksum method is quite similar across the various number of processes, although, from Table 7.4, Pranksum uses less time in every genomic region except 5′-UTR.

As discussed in section 6.3, we sort the input data by size, largest to smallest, such that the most computationally heavy jobs start before the less computationally heavy. Looking at the time it took finding all DhMRs of 5′-UTR regions using Rranksum, as shown in Table 7.4, and the lowest part of the Rranksum graph in Figure 7.11 as an example, we see that sorting the input data by size has worked as intended. The full **DMR Search** task took approximately 4000 seconds with three processes, which is the same as 66 minutes and 40 seconds. This is approximately the same as the biggest execution time in Table 7.4, which is 71m22s when finding DhMRs in 5′-UTR regions. This means that the whole pipeline uses approximately the same amount of time for running the biggest job, and running the whole task. If we did not sort the data files by size before distributing the jobs between the processes, we could not have been certain that the biggest jobs would start first. By parallelizing this task, we have had a 2.6x speed up, going from one process to three.

When looking at the memory consumption in Figure 7.12, we see that for Pranksum and Rranksum the memory consumption is fairly stable, where increasing the number of processes will linearly increase the memory consumption. The same is for Mranksum when the number of processes is smaller than 8. However, the memory consumption increase significantly more in Mranksum. That is because each process starts a new MATLAB engine, which will increase the memory consumption by a factor equal to the size of the memory allocation of one MATLAB engine.

Table 7.4: Time consumption of **DMR Search** for the three test methods: Pranksum, Mranksum, and Rranksum.

Genomic Region	**State**	**Execution Time**		
		Pranksum	**Mranksum**	**Rranksum**
TSS	**5mC**	0m2s	0m14s	2m48s
	5hmC	0m2s	0m14s	2m50s
TES	**5mC**	0m0.4s	0m10s	0m44s
	5hmC	0m0.4s	0m10s	0m44s
Gene Body	**5mC**	0m20s	0m27s	8m3s
	5hmC	0m21s	0m27s	7m52s
5′-UTR	**5mC**	3m33s	2m45s	68m40s
	5hmC	3m42s	3m5s	71m22s
Intergenic	**5mC**	0m19s	0m25s	6m44s
	5hmC	0m19s	0m25s	6m30s
Total	-	8m43s	8m26s	2h56m25s

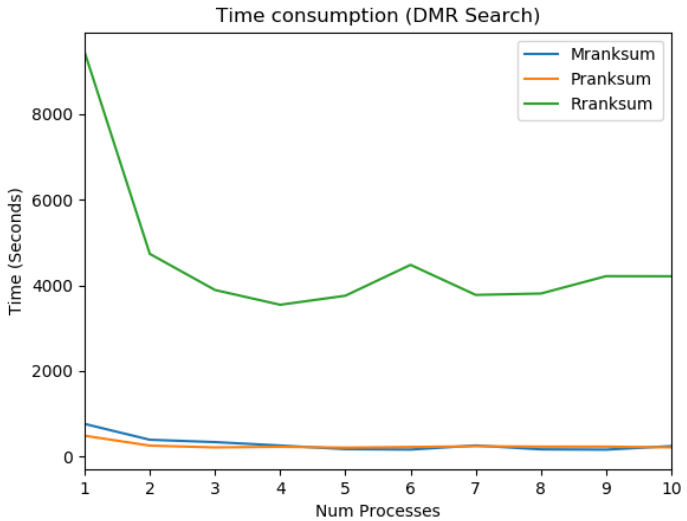

Figure 7.11: Time consumption of the **DMR Search** task with various number of processes.

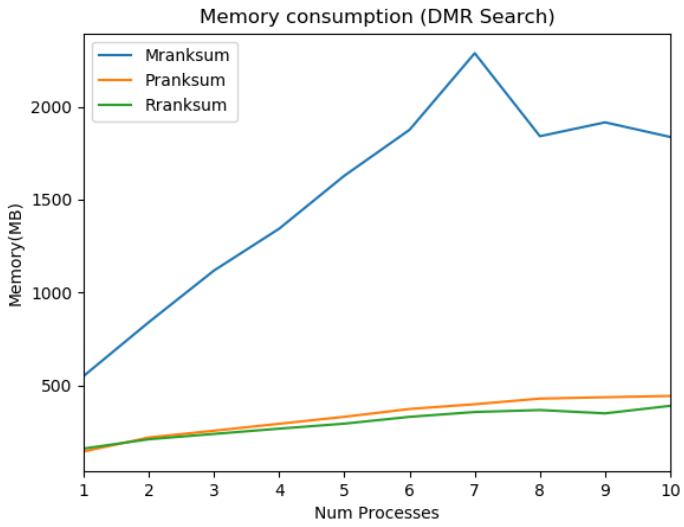

Figure 7.12: Memory consumption of the **DMR Search** task with various number of processes.

7.8 Applying Benjamini and Hochberg's FDR-Controlling Procedure When Detecting DMRs and DhMRs

In the current pipeline, we have only tested for DMRs/DhMRs using overlapping MRs where missing values were imputed by zeros, without a statistical correction for multiple comparisons. In this section, we run the **DMR Search** task, and apply the Benjamini-Hochberg procedure to the p-values that is obtained from Pranksum method, where overlapping MRs are considered and missing values were imputed by zero. When we applied a false discovery rate of $\alpha = 0.2$, none of the p-values was considered significant. In Figure 7.13, we illustrate why there were no significant results. This figure shows all p-values of 5mC-TSS overlapping MRs, calculated using the Pranksum method, as well as the threshold line, at each rank (i), with $\alpha = 0.2$. The Benjamini-Hochberg Procedure, as described in section 2.3.4, is the procedure of defining a threshold that can be used to declare tests as significant or not at level α. The threshold line, or Benjamini-Hochberg critical values, is calculated as $H_i = \frac{i}{m}\alpha$, where m is the total number of tests, shown as the orange line in the figure, for $i = 1, 2, ..., 687$. If there were any significant results, we should have seen one or more p-values below the threshold line. This does not mean we can conclude that there is no significant difference between the KO condition sample and the WT condition sample, because an inevitable byproduct of multiple comparisons corrections is that the number of false negatives is increased.

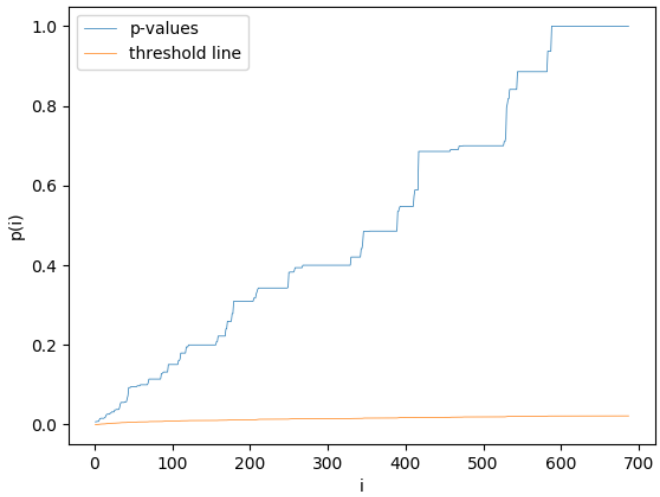

Figure 7.13: The sorted p-values (blue) found using the Pranksum method on overlapping MRs in 5mC TSS regions, and the threshold line (orange) at level $\alpha = 0.2$.

7.9 Experimental Analysis

Based on the exported results from the pipeline, users can perform their own analysis. Further exploring the pipeline results may reveal more hidden information in the datasets such as looking at the distribution of the number of 5mC/5hmC sites in overlapping MRs and DMRs/DhMRs. To provide an example of such analysis, we created two small Python scripts to investigate the distribution of the number of 5mC/5hmC sites in the overlapping MRs and DMRs/DhMRs found by the various test methods (Pranksum, Mranksum, and Rranksum). These results are shown in Figures 7.14, 7.15, and 7.16. Here, we do not differ between the different genomic regions, but add the numbers from each genomic region up. For example, from these figures, we can see that due to the minimum number of consecutive 5mC/5hmC sites in each MR is set to three in TSS and TES MRs, while it, in gene body, intergenic, and 5'-UTR regions is set to five, there are few overlapping MRs, DMRs, and DhMRs with less than five 5mC/5hmC sites.

7.10 Pipeline Run On Public HMST-Seq Data

As previously mentioned, the mouse dataset used above is confidential. We have, therefore, conducted a test run of the analysis pipeline using a public dataset generated by the HMST-Seq method as well. This data is obtained from the paper by F. Gao et al. [10], and consist of two HCC cell lines (97L and LM6 cells), and a non-HCC sample. Because the pipeline-input data must be in a certain format, we first had to remove some columns and change the order of the remaining columns after downloading the data. We

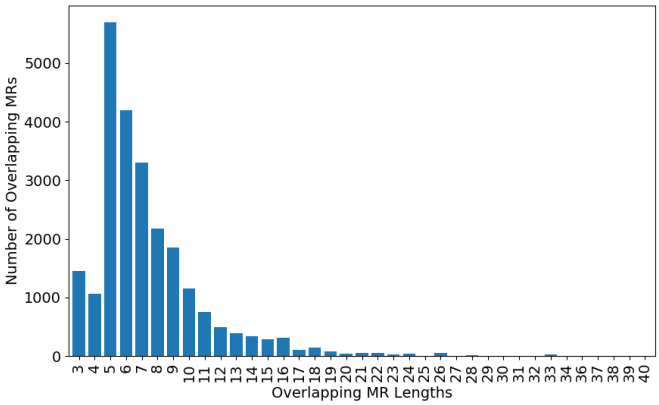

Figure 7.14: Distribution of overlapping MRs across various lengths.

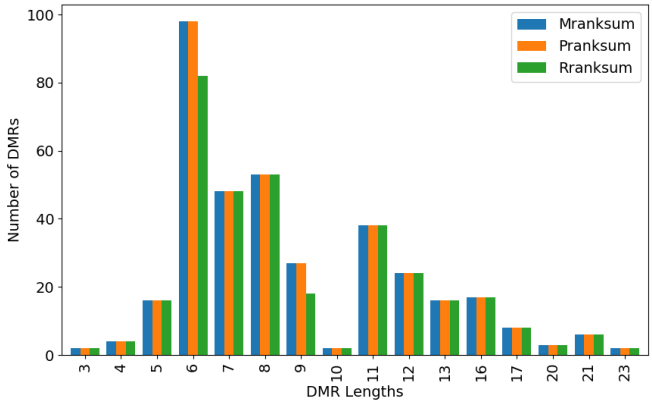

Figure 7.15: Distribution of DMRs found by Mranksum, Pranksum, and Rranksum across various lengths.

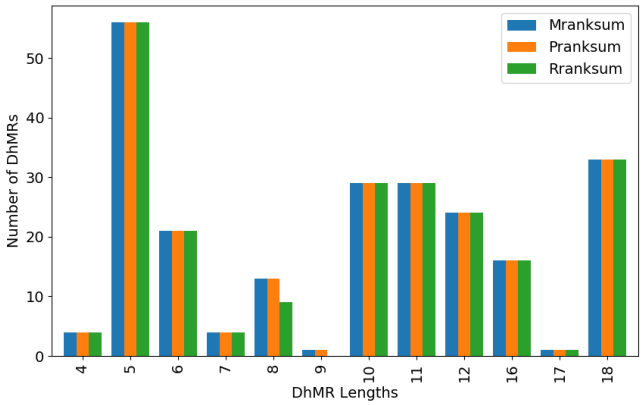

Figure 7.16: Distribution of DhMRs found by Mranksum, Pranksum, and Rranksum across various lengths.

did this using a simple awk script:

```
awk '{print $1 "\t" $2 "\t" $8 "\t" $4 "\t" $3}' %j > %i
```

where *%j* and *%i* is the input file name and output file name, respectively. This was done for all three samples.

Since the main purpose of the thesis is to illustrate the functions and features of the newly developed HMST-Seq-Analyzer package, we only applied the pipeline on chromosome 1 of this data as well. We used the following grep command to obtain just the chromosome 1 data:

```
grep -P "^chr1\t" %j > %i
```

where *%j* and *%i* are input and output file names, respectively.

For reference genome, we used the hg19 human assembly, because the tag counts of the dataset have been aligned to the hg19 reference genome. To remove the unwanted chromosomes from the refFlat.txt.gz file, downloaded from the UCSC Genome Browser, we used the zgrep command as following:

```
zgrep -P "\tchr(\d+|M|X|Y)\t" refFlat.txt.gz > hg19.refFlat.txt
```

We ran the following grep command on the chromosome sizes file to remove the unwanted chromosomes:

```
grep -P "chr(\d+|M|X|Y)\t" hg19.chrom.sizes > hg19.chrom.sizes.clear
```

followed by a sort command to sort the rows based on the chromosome name:

```
sort -k 1,1 -V hg19.chrom.sizes.clear > hg19.chrom.sizes.clear.sorted
```

where the argument *-V* means natural sort, such that chr1 comes before chr2, and chrX comes before chrY, etc. However, because chrM comes before chrX when sorting, we had to change this manually.

By running the full pipeline with all default input arguments and this data as input, we obtained the figures shown in Figure 7.17, 7.18, 7.19, and 7.20. These results show a generally high amount of hypo DhMRs, especially in TSS regions, when comparing the HCC cell lines with the non-HCC sample. 5mC levels around TSS seem to be slightly higher in the HCC samples than in the non-HCC sample. In the paper by Y. Zhu et al. [44], they found an uneven distribution of 5mC and 5hmC in TSS regions of tumor tissues when compared with normal tissues. They also observed that 5mC levels of tumor tissues were generally higher than in normal tissues, and the opposite for 5hmC levels. Our results suggest the same, where approximately 35% of MRs in TSS regions are DhMRs, while only about 5% are DMRs. In Figure 7.18, the number of significantly modified 5mC sites is lower for the non-HCC sample in all genomic regions than that of the HCC ones. Thus, our new pipeline reproduce the results of the original publication.

7.11 Testing on WGBS Data and Comparing Results with methylKit

In this section, we test the pipeline on public WGBS data from the ENCODE project, and compare the results with that from another differential

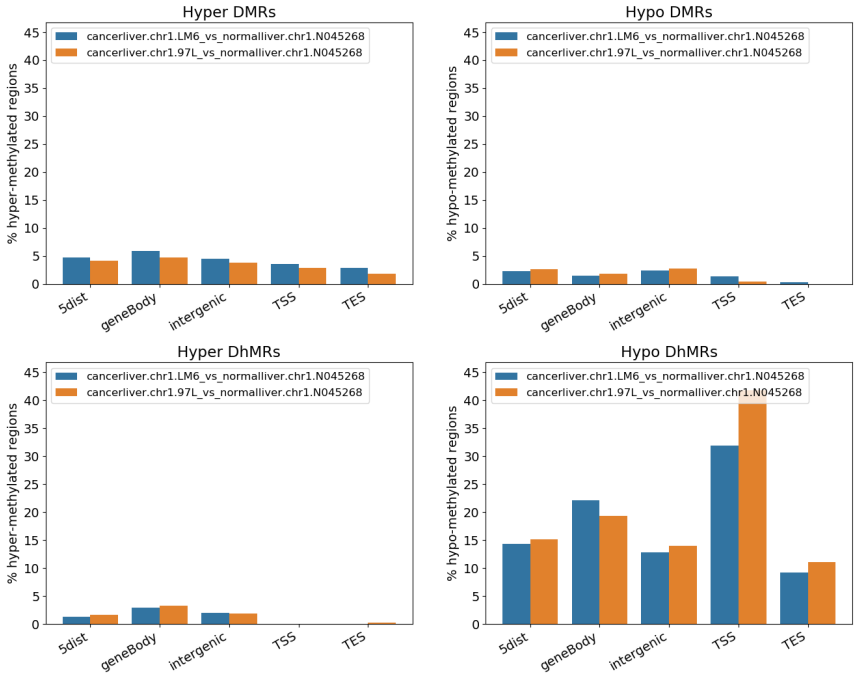

Figure 7.17: The distribution of hypo and hyper DMRs and DhMRs of the two HCC cell lines, 97L and LM6, versus a non-HCC sample, within different genomic regions.

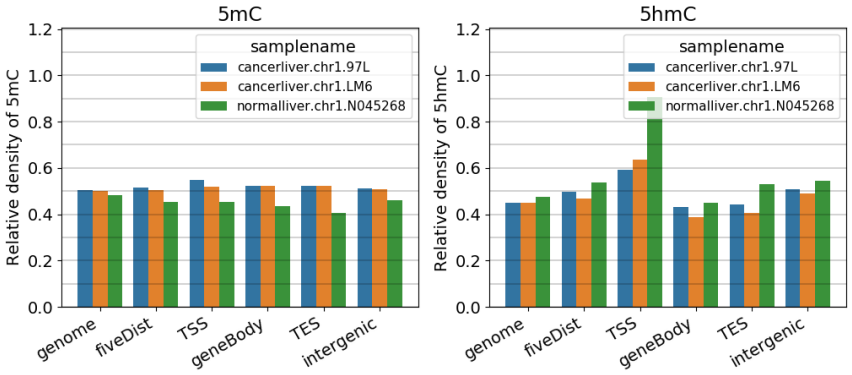

Figure 7.18: The realtive density of significantly modified sites in MRs of the two HCC cell lines, 97L and LM6, and one non-HCC sample, within the various genomic regions.

methylation analysis tool - methylKit [45]. methylKit is an R package for DNA methylation analysis and annotation from high-throughput bisulfite sequencing. It can carry out operations such as differential methylation analysis, sample clustering and annotation, and visualization of DNA methylation events.

We used a human dataset with one test sample (GM12878: human lymphoblastoid cell line) and one control sample (H1: human embryonic stem cell line), which was directly downloaded from the Encyclopedia of DNA Elements (ENCODE) [46]. When downloading these datasets, we

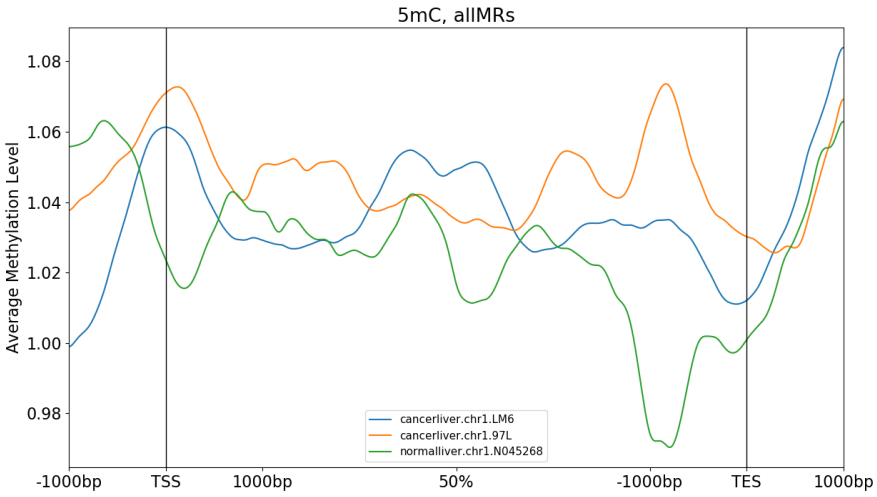

Figure 7.19: The distribution of 5mC levels of MRs of the two HCC cell lines, 97L and LM6, and one non-HCC sample, in the combined TSS, gene body, and TES regions.

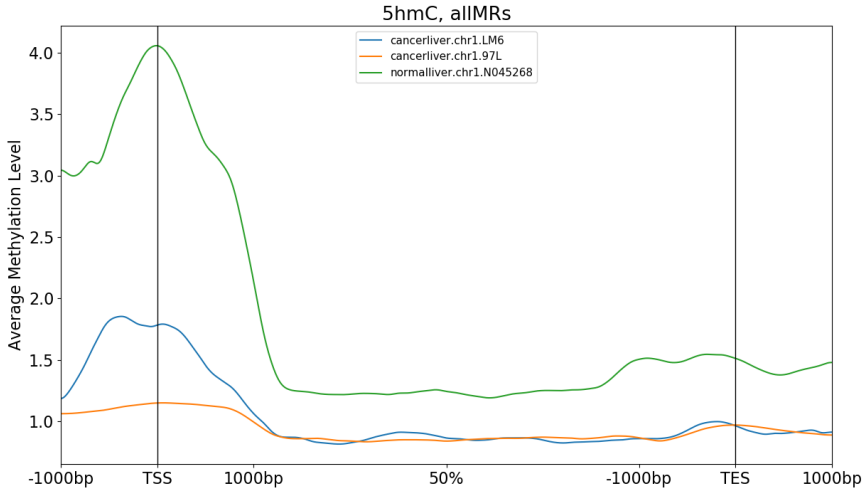

Figure 7.20: The distribution of 5hmC levels of MRs of the two HCC cell lines, 97L and LM6, and one non-HCC sample, in the combined TSS, gene body, and TES regions.

chose the BED-formatted CpG methylation state files. ENCODE provide human data aligned to hg38, so we downloaded the hg38 reference genome and chromosome sizes files and prepared them as described in the previous section. Since WGBS data is huge, we split it into subsets based on chromosome names and ran the pipeline in parallel. Here, to save the CPU time in computation, we set 5'-UTR regions from 10000 bp to 50000 bp of TSS in the pipeline. We ran the pipeline by creating batch scripts for every chromosome and submitting all jobs, such that they could run in parallel. After all jobs finished, we collected the sites of every DMR for all the genomic regions: TSS, gene body, TES, intergenic, and 5'-UTR and kept all unique sites for each chromosome. When collecting the DMR sites, we

created two sets: one set where missing sites were removed, and the other set where we kept all DMR sites.

For methylKit, the analysis was also done parallel across all chromosomes. Here, only CG sites that are common in both samples are considered. All unique differentially methylated sites from methylKit are collected and compared to that from the HMST-Seq-Analyzer. This comparison is shown in the two Venn diagrams of Figure11 423 519 7.21 and 7.22. Both figures show that approximately 97% of the differentially methylated sites found by methylKit overlap with the DMR sites found by our pipeline. This result indicates that the HMST-Seq-Analyzer is not affected by the missing values and that the results are robust in differential methylation analysis.

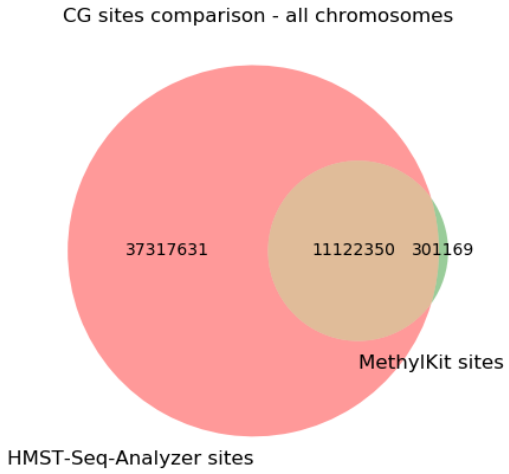

Figure 7.21: Venn diagram comparing common sites of DMRs obtained by the HMST-Seq-Analyzer with differentially methylated sites obtained by methylKit.

Since our HMST-Seq-Analyzer's main analysis consists of finding DMRs, we also compared our DMRs with DMRs of methylKit. methylKit has the option of summarizing the methylation information over tiling windows rather than doing base-pair resolution analysis. We, therefore, tiled the samples with window size equal to 1000 bp (default) but changed the step-size to 50 bp to save computing time. The methylation information in each tile was summarized accordingly. We then did differential methylation analysis on the tiled windows and merged all overlapping regions using the merge feature of BEDTools. The DMRs found by our pipeline were first combined over all the genomic regions and then merged using BEDTools, such that overlapping DMRs were combined. The number of unique, merged DMRs and differentially methylated tiled windows is 11101 for HMST-Seq-Analyzer, and 230167 for methylKit, respectively. We will, for the rest of this section, refer to the differentially methylated tiled windows of methylKit as DMRs as well.

Since we only consider tiled windows of methylKit of length 1000bp,

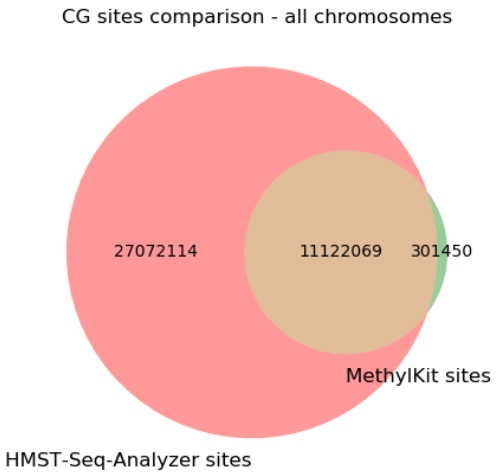

Figure 7.22: Venn diagram comparing intersecting sites of DMRs obtained by the HMST-Seq-Analyzer with differentially methylated sites obtained by methylKit.

the DMRs of methylKit are generally shorter than DMRs from HMST-Seq-Analyzer. However, since we merge overlapping DMRs, we have made a comparison of the DMR length distribution, such that we will know whether we should check for overlaps based on the HMST-Seq-Analyzer DMRs or the methylKit DMRs. By doing this, we will get an idea of which of the DMR sets are generally smaller than the other. We created two histograms to illustrate the DMR length distribution, which can be seen in Figure 7.23 and 7.24. From these figures, we see that the DMR lengths of HMST-Seq-Analyzer are generally longer than that of methylKit, where the median for HMST-Seq-Analyzer is approximately 100000, while it for methylKit is between 1000 and 10000. This supports our initial claim: that the HMST-Seq-Analyser's DMRs are longer than that of methylKit. Because of this, we can check overlaps based on methylKit's DMRs.

Using BEDTools intersect, when the fraction of overlap is defined from 10% to 100%, the number of DMRs overlapping between the two tests is between 10768 and 9643, as can be seen in Figure 7.26 and 7.25. Comparing this to the total number of merged DMRs, which is 11101 and 230167 for HMST-Seq-Analyzer and methylKit, respectively, this means that around 80% of our predicted DMRs have methylKit DMRs overlapping 100% with it, and that over 90% of our predicted DMRs have methylKit DMRs overlapping at least 10% with it. When using methylkit tiled windows, genome-wide data is used. In HMST-Seq-Analyzer, only data from the five predefined genomic regions (TSS, TES, gene body, intergenic, and 5'-UTR) is used. Knowing this, and the fact that we used relatively small 5'-UTR regions in the current comparison (10000 bp to 50000 bp upstream of TSS), we can say that if HMST-Seq-Analyzer were to consider genome-wide data also, then a much higher overlapping between the two results is to be expected.

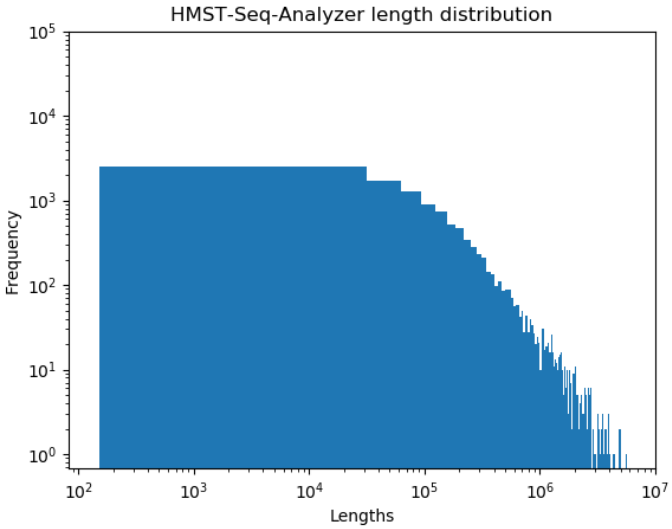

Figure 7.23: Histogram showing the distribution of unique, merged DMR lengths found by HMST-Seq-Analyzer.

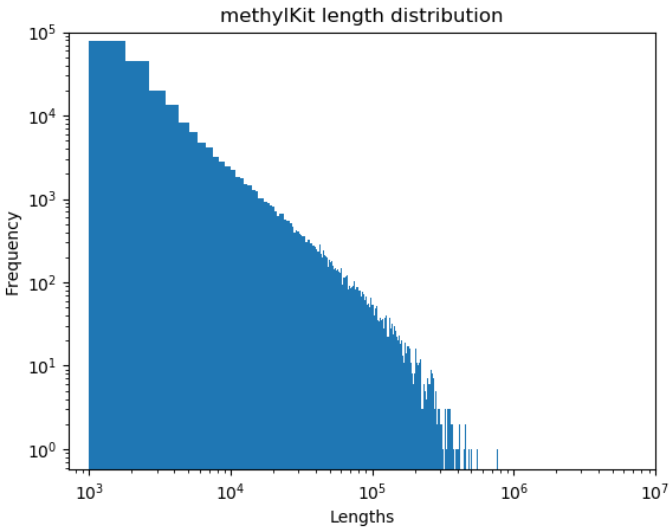

Figure 7.24: Histogram showing the distribution of unique, merged DMR lengths found by methylKit.

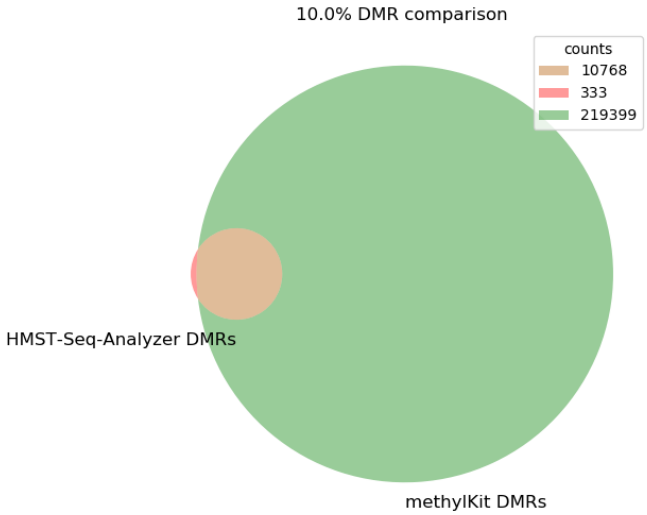

Figure 7.25: Venn diagram showing the number of overlaps where at least 10% of the merged DMRs from methylKit overlap with HMST-Seq-Analyzer merged DMRs (yellow), the number of merged DMRs from HMST-Seq-Analyzer not overlapping (red), and the number of merged DMRs from methylKit not overlapping (green).

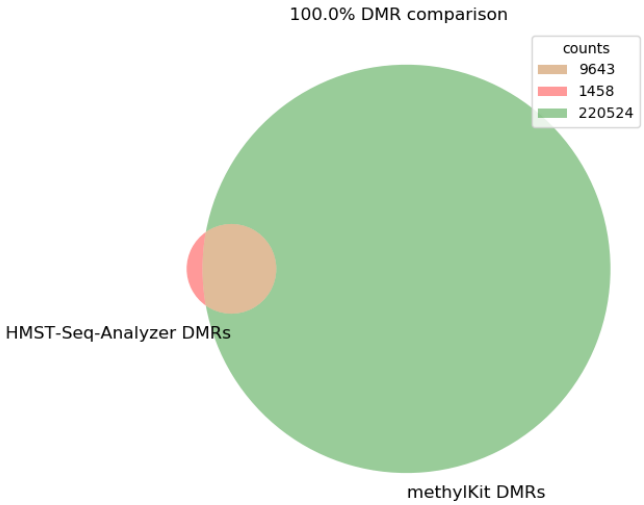

Figure 7.26: Venn diagram showing the number of overlaps where 100% of the merged DMRs from methylKit overlaps with HMST-Seq-Analyzer merged DMRs (yellow), the number of merged DMRs from HMST-Seq-Analyzer not overlapping (red), and the number of merged DMRs from methylKit not overlapping (green).

Chapter 8

Conclusion and Future Work

8.1 Conclusion

There are several tools available for differential methylation analysis [47]. Most of them are created for data generated from BS-Seq, and can, therefore, not differentiate 5mC and 5hmC. Some tools have been made that are able to analyze 5mC and 5hmC separately, but they are difficult to use for analyzing both datasets simultaneously. We have, therefore, developed a new computational analysis pipeline by considering a proper statistical method to analyze DNA methylation data generated by the HMST-Seq technique. In addition, the pipeline can also handle methylation data produced by bisulfite sequencing. A part of the analysis is to find differentially methylated and hydroxymethylated regions in genomic regions, such as TSS, TES, gene body, intergenic, 5'-UTR, and enhancer. The analysis pipeline produces figures showing the distribution of DMRs and DhMRs, the distribution of 5mC and 5hmC MRs in TSS, gene body, TES, and enhancers as well as the distribution of significantly modified 5mC and 5hmC sites.

The analysis pipeline was developed as a Python package consisting of eight main tasks, that works for datasets generated from both mouse and human samples. Users can set various input arguments as preferred, making the pipeline open for various experimental conditions. To find DMRs and DhMRs, there are three types of methods in the pipeline. Though all three methods are related to the Wilcoxon rank-sum test, it is implemented by different computational algorithms. We compared the test methods, showing that our Pranksum method predicts the same DMRs and DhMRs as MATLAB's ranksum function, but having significantly lower memory consumption.

We have tested the new pipeline on two datasets produced by HMST-seq experiments. It successfully identified DMRs and DhMRs between the two conditions. Our predictions are consistent with the literature information that CG methylation in TSS regions is an important feature of gene regulation in mammals. It also supports the previous evidence in differential methylation analysis of cancer cell lines. The result of the current pipeline was also compared to that of methylKit. This comparison

shows that our pipeline recovered approximately 90% of the differentially methylated CG sites predicted by methylKit. Additionally, around 90% of our DMRs have differentially methylated tiled windows from methylKit. A much higher overlapping between HMST-Seq-Analyzer and methylKit will be expected if our pipeline considers the same genome-wide data as methylKit does.

8.2 Limitations and Future Work

The three most computationally heavy tasks of the pipeline are the tasks that find MRs, overlaps MRs between two conditions and predict DMRs and DhMRs. Even though there is an option for parallelizing these tasks, they require a large amount of computational time when very large datasets are input. Some improvements to reduce the time consumption should be done in the future. In this section, we present some suggestions for improvements and possible future additions.

8.2.1 Possible Future Improvement in HMST-Seq-Analyzer

The two tasks **Find MRs** and **Preparation for DMR Search** finds genomic regions with clusters of 5mC and 5hmC sites, and discovers overlaps between two conditions based on the genome sites, respectively. Both the 5mC and the 5hmC levels are calculated from the same genome sites. Therefore, the two sets contain the exact same genome sites, but have different methylation levels. Because the methylation levels is not important to the aforementioned tasks, and both datasets contain the exact same genome sites, the two tasks (**Find MRs** and **Preparation for DMR Search**) could be done only once, instead of once for each of the 5mC and 5hmC dataset. This could potentially half the execution time of the two tasks when using HMST-Seq datasets.

Currently, the HMST-Seq-pipeline only supports pair-wise comparison between two samples. However, in biology, it is very interesting to study differential methylation between two groups of samples, or even multiple groups. This would be a very useful new feature in the future version of HMST-Seq-Analyzer.

Though our results show a good overlap of differentially methylated CG (DMCG) sites between methylKit and HMST-Seq-Analyzer, the DMCG sites of HMST-Seq-Analyzer were taken from predicted DMRs. In the future, a direct comparison of individual CG sites may be added in HMST-Seq-Analyzer. Especially, a graphical user interface should be implemented in the pipeline, which will make it easier for users to apply the pipeline on various datasets.

Bibliography

[1] T. A. Brown, *Genomes 2nd edition*, English. BIOS Scientific Publishers, 2002.

[2] D. R. Morris and A. P. Geballe, "Upstream open reading frames as regulators of mrna translation", *Molecular and Cellular Biology*, vol. 20, no. 23, pp. 8635–8642, 2000. DOI: 10.1128/MCB.20.23.8635-8642.2000.

[3] Y. Shen, F. Yue, D. F. McCleary, Z. Ye, L. Edsall, S. Kuan, U. Wagner, J. Dixon, L. Lee, V. V. Lobanenkov and B. Ren, "A map of the cis-regulatory sequences in the mouse genome", *Nature*, vol. 488, no. 2, p. 116, 7409 Aug. 2012. DOI: 10.1038/nature11243.

[4] L. A. Pennacchio, W. Bickmore, A. Dean, M. A. Nobrega and G. Bejerano, "Enhancers: Five essential questions", *Nature reviews. Genetics*, vol. 14, no. 4, pp. 288–295, Apr. 2013. DOI: 10.1038/nrg3458.

[5] J. Tost, "Dna methylation: An introduction to the biology and the disease-associated changes of a promising biomarker", in *DNA Methylation: Methods and Protocols*. Totowa, NJ: Humana Press, 2009, pp. 3–20. DOI: 10.1007/978-1-59745-522-0_1.

[6] H. Xu, "Differential methylation analysis with next-generation sequencing", in *Next Generation Sequencing in Cancer Research, Volume 2: From Basepairs to Bedsides*, W. Wu and H. Choudhry, Eds. Cham: Springer International Publishing, 2015, pp. 229–238. DOI: 10.1007/978-3-319-15811-2_14.

[7] L. A. Vivanco, *Phenotype*, Jan. 2018. [Online]. Available: http://www.oxfordreference.com/view/10.1093/acref/9780191836688.001.0001/acref-9780191836688-e-270 (visited on 11/04/2019).

[8] Y. Xia, J. Wang, H. Luo, G. Zhaowei, X. Han, J. Zhang, X.-J. Huang, Y. Yao, H. Lu, N. Yi, B. Zhou, Z. Lin, B. Wen, H. Yang, X. Zhang, J. Wang and F. Gao, "Integrated detection of both 5-mc and 5-hmc by high-throughput tag sequencing technology highlights methylation reprogramming of bivalent genes during cellular differentiation", *Epigenetics : official journal of the DNA Methylation Society*, vol. 8, Mar. 2013. DOI: 10.4161/epi.24280.

[9] Y. Chen, B. Pal, J. E. Visvader and G. K. Smyth, "Differential methylation analysis of reduced representation bisulfite sequencing experiments using edger [version 1; referees: 2 approved, 1 approved

with reservations]", *F1000Research*, vol. 6, no. 2055, 2017. DOI: 10.12688/f1000research.13196.1.

[10] F. Gao, Y. Xia, J. Wang, Z. Lin, Y. Ou, X. Liu, W. Liu, B. Zhou, H. Luo, B. Zhou, B. Wen, X. Zhang and J. Huang, "Integrated analyses of dna methylation and hydroxymethylation reveal tumor suppressive roles of ecm1, atf5, and eomes in human hepatocellular carcinoma", *Genome biology*, vol. 15, p. 533, Dec. 2014. DOI: 10.1186/s13059-014-0533-9.

[11] F. Wang, N. Zhang, J. Wang, H. Wu and X. Zheng, "Tumor purity and differential methylation in cancer epigenomics", *Briefings in Functional Genomics*, vol. 15, no. 6, pp. 408–419, 2016. DOI: 10.1093/bfgp/elw016.

[12] D. N. Ayyala, D. E. Frankhouser, J.-O. Ganbat, G. Marcucci, R. Bundschuh, P. Yan and S. Lin, "Statistical methods for detecting differentially methylated regions based on methylcap-seq data", *Briefings in bioinformatics*, vol. 17, no. 6, pp. 926–937, Nov. 2016. DOI: 10.1093/bib/bbv089.

[13] S. Kriaucionis and N. Heintz, "The nuclear dna base 5-hydroxymethy lcytosine is present in purkinje neurons and the brain", *Science*, vol. 324, no. 5929, pp. 929–930, May 2009. DOI: 10.1126/science.1169786.

[14] C. R. Pelz, M. Kulesz-Martin, G. Bagby and R. C. Sears, "Global rank-invariant set normalization (grsn) to reduce systematic distortions in microarray data", *BMC Bioinformatics*, vol. 9, no. 1, p. 520, Dec. 2008. DOI: 10.1186/1471-2105-9-520.

[15] B. Bolstad, R. Irizarry, M. Åstrand and T. Speed, "A comparison of normalization methods for high density oligonucleotide array data based on variance and bias", *Bioinformatics*, vol. 19, no. 2, pp. 185–193, 2003. DOI: 10.1093/bioinformatics/19.2.185.

[16] J. L. Devore and K. N. Berk, "Tests of hypotheses based on a single sample", in *Modern Mathematical Statistics with Applications*. New York, NY: Springer New York, 2012, pp. 425–483. DOI: 10.1007/978-1-4614-0391-3_9.

[17] R. Chin and B. Y. Lee, "Chapter 15 - analysis of data", pp. 325 –359, 2008. DOI: https://doi.org/10.1016/B978-0-12-373695-6.00015-6.

[18] C. Wild and G. A. F. Seber, *Chance Encounters: A First Course in Data Analysis and Inference*. John Wiley & Sons, New York, 1999.

[19] C. Wild, *The wilcoxon rank-sum test*, 1997. [Online]. Available: https://www.stat.auckland.ac.nz/\%7Ewild/ChanceEnc/Ch10.wilcoxon.pdf (visited on 17/10/2019).

[20] C.-C. Günther, Ø. Bakke, H. Rue and M. Langaas, *Statistical hypothesis testing for categorical data using enumeration in the presence of nuisance parameters*, Trondheim, 2009.

[21] S.-Y. Chen, Z. Feng and X. Yi, "A general introduction to adjustment for multiple comparisons", *Journal of thoracic disease*, vol. 9, no. 6, pp. 1725–1729, Jun. 2017. [Online]. Available: https://www.ncbi.nlm.nih.gov/pubmed/28740688 (visited on 29/10/2019).

[22] Y. Benjamini and Y. Hochberg, "Controlling the false discovery rate - a practical and powerful approach to multiple testing", *J. Royal Statist. Soc., Series B*, vol. 57, pp. 289 –300, Nov. 1995. DOI: 10.2307/2346101.

[23] J. H. McDonald, *Handbook of Biological Statistics*. Sparky House Publishing Baltimore, MD, 2009, vol. 2.

[24] R. Fisher, S. Perkins, A. Walker and E. Wolfart., *Gaussian smoothing*, 2003. [Online]. Available: https://homepages.inf.ed.ac.uk/rbf/HIPR2/gsmooth.htm (visited on 27/07/2019).

[25] W. J. Kent, C. W. Sugnet, T. S. Furey, K. M. Roskin, T. H. Pringle, A. M. Zahler and D. Haussler, *The human genome browser at ucsc*, Jun. 2002. [Online]. Available: http://genome.ucsc.edu/ (visited on 15/10/2019).

[26] National Center for Biotechnology Information (NCBI), "Bethesda (md): National library of medicine (us)", 1988. [Online]. Available: https://www.ncbi.nlm.nih.gov/ (visited on 15/10/2019).

[27] F. Madeira, Y. m. Park, J. Lee, N. Buso, T. Gur, N. Madhusoodanan, P. Basutkar, A. R. N. Tivey, S. C. Potter, R. D. Finn and R. Lopez, "The EMBL-EBI search and sequence analysis tools APIs in 2019", *Nucleic Acids Research*, vol. 47, no. W1, W636–W641, Apr. 2019. DOI: 10.1093/nar/gkz268.

[28] G. Van Rossum and F. L. Drake Jr, *Python reference manual*, Virginia, USA, 2001. [Online]. Available: http://www.python.org (visited on 27/07/2019).

[29] W. McKinney, "Data structures for statistical computing in python", *Proceedings of the 9th Python in Science Conference*, vol. 445, pp. 51–56, Jan. 2010.

[30] T. E. Oliphant, *Guide to NumPy*. Trelgol Publishing USA, 2006, vol. 1.

[31] J. D. Hunter, "Matplotlib: A 2d graphics environment", *Computing in science & engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[32] M. Waskom, O. Botvinnik, P. Hobson, J. B. Cole, Y. Halchenko, S. Hoyer, A. Miles, T. Augspurger, T. Yarkoni, T. Megies, L. P. Coelho, D. Wehner, cynddl, E. Ziegler, diego0020, Y. V. Zaytsev, T. Hoppe, S. Seabold, P. Cloud, M. Koskinen, K. Meyer, A. Qalieh and D. Allan, "Seaborn: V0.5.0 (november 2014)", Nov. 2014. DOI: 10.5281/zenodo.12710.

[33] E. Jones, T. Oliphant, P. Peterson *et al.*, *SciPy: Open source scientific tools for Python*, 2001–. [Online]. Available: http://www.scipy.org/ (visited on 26/07/2019).

[34] Anaconda Software Distribution, *Computer software. vers. 2-2.4.0. anaconda*, Nov. 2016. [Online]. Available: https : / / anaconda . com (visited on 27/07/2019).

[35] A. R. Quinlan and I. M. Hall, "BEDTools: a flexible suite of utilities for comparing genomic features", *Bioinformatics*, vol. 26, no. 6, pp. 841–842, Jan. 2010. DOI: 10.1093/bioinformatics/btq033.

[36] MATLAB, Statistics and Machine Learning Toolbox, *Version 9.2.0.556 344 (r2017a)*, Natick, Massachusetts, 2017.

[37] R Core Team, *R: A language and environment for statistical computing*, Vienna, Austria: R Foundation for Statistical Computing, 2017. [Online]. Available: https : / / www . R - project . org/ (visited on 26/07/2019).

[38] L. He and G. J. Hannon, "Microrнas: Small rnas with a big role in gene regulation", *Nature Reviews Genetics*, vol. 5, no. 7, pp. 522–531, 2004. DOI: 10.1038/nrg1379.

[39] J. Wang, X. Lan, P.-Y. Hsu, H.-K. Hsu, K. Huang, J. Parvin, T. H.-M. Huang and V. X. Jin, "Genome-wide analysis uncovers high frequency, strong differential chromosomal interactions and their associated epigenetic patterns in e2-mediated gene regulation", *BMC genomics*, vol. 14, pp. 70–70, Jan. 2013. DOI: 10.1186/1471-2164-14-70.

[40] B. Tang, Y. Zhou, C.-M. Wang, T. H. M. Huang and V. X. Jin, "Integration of dna methylation and gene transcription across nineteen cell types reveals cell type-specific and genomic region-dependent regulatory patterns", *Scientific Reports*, vol. 7, no. 1, p. 3626, 2017. DOI: 10.1038/s41598-017-03837-z.

[41] R. Bergmann, J. Ludbrook and W. P.J. M. Spooren, "Different outcomes of the wilcoxon-mann-whitney test from different statistics packages", *The American Statistician*, vol. 54, no. 1, pp. 72–77, 2000. DOI: 10.1080/00031305.2000.10474513.

[42] B. Grüning, R. Dale, A. Sjödin, B. A. Chapman, J. Rowe, C. H. Tomkins-Tinch, R. Valieris, J. Köster and T. B. Team, "Bioconda: Sustainable and comprehensive software distribution for the life sciences", *Nature Methods*, vol. 15, no. 7, pp. 475–476, 2018. DOI: 10.1038/s41592-018-0046-7.

[43] P. A. Jones, "Functions of dna methylation: Islands, start sites, gene bodies and beyond", *Nature Reviews Genetics*, vol. 13, no. 7, pp. 484–492, 2012. DOI: 10.1038/nrg3230.

[44] Y. Zhu, H. Lu, D. Zhang, M. Li, X. Sun, L. Wan, D. Yu, Y. Tian, H. Jin, A. Lin, F. Gao and M. Lai, "Integrated analyses of multi-omics reveal global patterns of methylation and hydroxymethylation and screen the tumor suppressive roles of hadhb in colorectal cancer", *Clinical epigenetics*, vol. 10, pp. 30–30, 2018. DOI: 10.1186/s13148-018-0458-3.

[45] A. Akalin, M. Kormaksson, S. Li, F. E. Garrett Bakelman, M. E. Figueroa, A. Melnick and C. E. Mason, "Methylkit: A comprehensive r package for the analysis of genome-wide dna methylation profiles", *Genome Biology*, vol. 13, no. 10, R87, Oct. 2012. DOI: 10.1186/gb-2012-13-10-r87.

[46] C. A. Davis, B. C. Hitz, C. A. Sloan, E. T. Chan, J. M. Davidson, I. Gabdank, J. A. Hilton, K. Jain, U. K. Baymuradov, A. K. Narayanan, K. C. Onate, K. Graham, S. R. Miyasato, T. R. Dreszer, J. S. Strattan, O. Jolanki, F. Y. Tanaka and J. M. Cherry, "The encyclopedia of dna elements (encode): Data portal update", *Nucleic acids research*, vol. 46, no. D1, pp. D794–D801, Jan. 2018. DOI: 10.1093/nar/gkx1081.

[47] A. E. Teschendorff and C. L. Relton, "Statistical and integrative system-level analysis of dna methylation data", *Nature Reviews Genetics*, vol. 19, 129 EP –, Nov. 2017, Review Article. [Online]. Available: https : / / doi . org / 10 . 1038 / nrg . 2017 . 86 (visited on 15/11/2019).