Methpipe Manual

Song Qiang

July 7, 2011

The Methpipe software package is a comprehensive tool chain for analyzing whole genome bisulfite sequencing data (BS-Seq). This documentation will guide you step by step how to perform the data analysis in a BS-Seq project with Methpipe. To facilitate understanding of the work flow, we divide the analysis procedure into four steps: (1) Pre-mapping processing includes assessing read quality and trimming adapters (if users use rmapbs for mapping, then trimming adapters as a separate step is not necessary); (2) Mapping sequenced reads to a reference genome; (3) Analyzing methylation level at a single site (CpG sites or non-CpG cytosines). (4) Higher level analysis includes identifying scores profiles (differential methylation or allelic methylation) and methylation characteristic regions (hypomethylated regions or differentially methylated regions).

Fig. **??** shows detailed work flow of analysis with Methpipe. Processes and corresponding tools are shown with rectangles: incoming and outgoing arrows represent the number of input and output files respectively. Files with intermediate input or output data are shown as parallelograms and files with the final results are shown as ovals. The order of processes is important.

We will use provided test data as an example how to run our tools. Suppose that in a project you would like to analyze methylomes of two cell types, ESC and NHFF. There are paired-end sequenced reads for ESC and single-end reads for NHFF:

```
test_ESC_1.fastq, test_ESC_2.fastq, test_NHFF.fastq
```

We will discuss tools in the same order as they appear in Fig. **??**. For each tool, we will describe what it does, the tool's options, and input/output formats. Most of the tools use files in BED format, therefore we will provide BED format now, and later only refer to it by the name.

BED format:

- reference name <string>
- start position within reference <integer>: starts with 0
- end position within reference <integer>: starts with 0
- name <string>: depends on the file (read ID, C– content, HMR ID, DMR ID, etc.)
- score <float>: dependes on the file (number of mismatches, methylation level, score of HMR or DMR, etc.)
- strand <string>

We will start with the tools for pre-mapping analysis (not shown in the workflow).

1 Pre-mapping processing

Before mapping sequenced reads to a reference genome, we need to pre-process the raw read sequences. In particular, we need to trim adapter sequences retained in the 3' end of raw reads. Further, we may be interested in examining the quality of reads in our library and visualizing raw reads in UCSC Genome Browser.

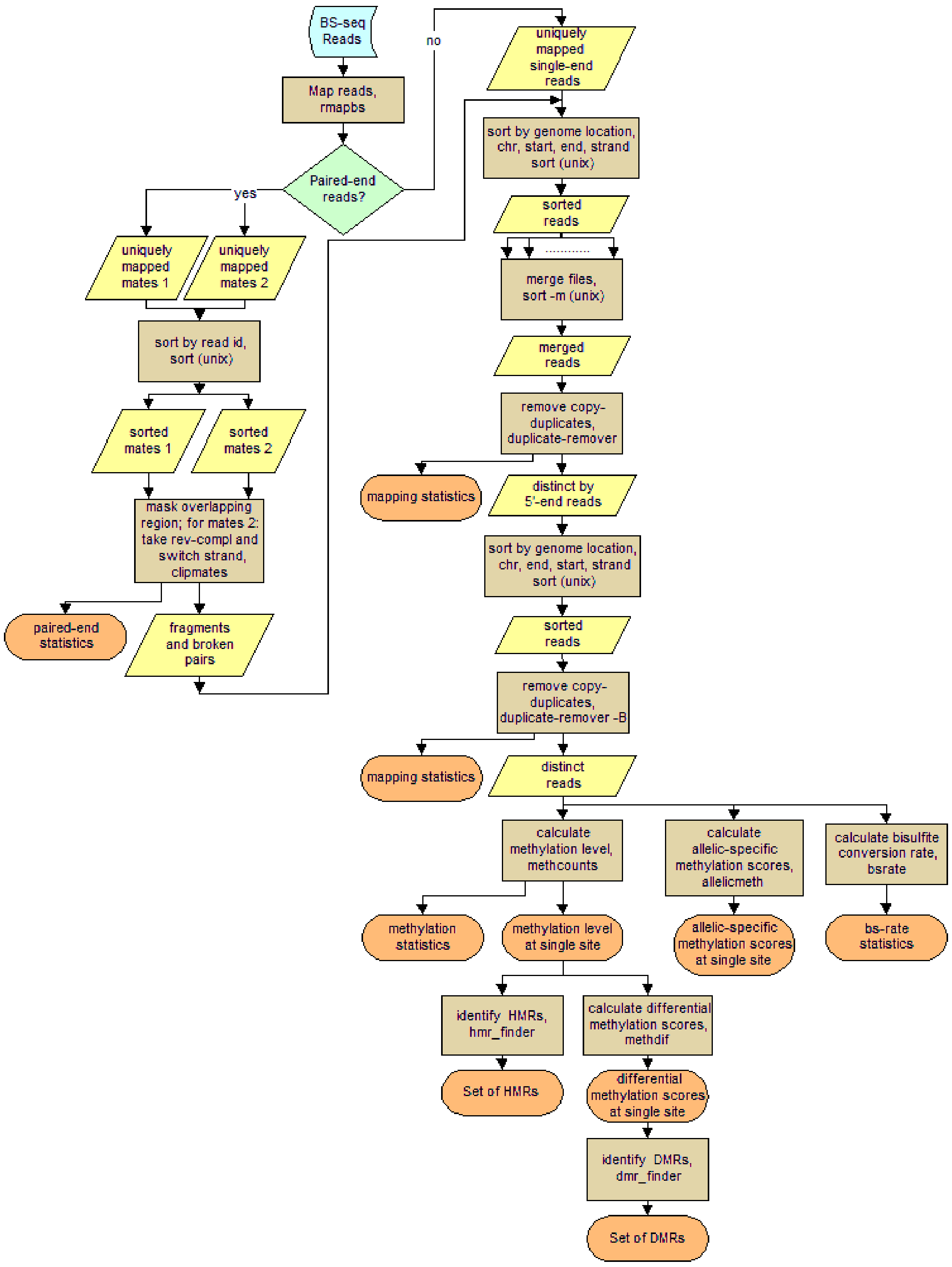

BS-seq Reads
Map reads, rmapbs
Paired-end reads?
yes
no
uniquely mapped mates 1
uniquely mapped mates 2
sort by read id, sort (unix)
sorted mates 1
sorted mates 2
mask overlapping region; for mates 2: take rev-compl and switch strand, clipmates
paired-end statistics
fragments and broken pairs
uniquely mapped single-end reads
sort by genome location, chr, start, end, strand sort (unix)
sorted reads
merge files, sort -m (unix)
merged reads
remove copy-duplicates, duplicate-remover
mapping statistics
distinct by 5'-end reads
sort by genome location, chr, end, start, strand sort (unix)
sorted reads
remove copy-duplicates, duplicate-remover -B
mapping statistics
distinct reads
calculate methylation level, methcounts
calculate allelic-specific methylation scores, allelicmeth
calculate bisulfite conversion rate, bsrate
methylation statistics
methylation level at single site
allelic-specific methylation scores at single site
bs-rate statistics
identify HMRs, hmr_finder
calculate differential methylation scores, methdif
Set of HMRs
differential methylation scores at single site
identify DMRs, dmr_finder
Set of DMRs

Figure 1: Methpipe's Workflow

1.1 Trim adapters

As the length of sequenced reads has been increasing all the time, it is possible there are adapter sequences in the 3' end of some reads. These retained adapter sequences affect the mappability of the reads. Even if the reads are somehow mapped to the reference genome, the adapter sequences may introduce bias to our estimate of methylation frequency. Therefore it is necessary to trim these retained adapter sequences.

The program **trim-adapter** is used to trim adapter sequences, if any, from the 3' end of raw reads. Suppose the adapter sequence is *GATCGGAAGAGCGGTTCAGCAGGAATGCCGAGACCGATCTCGTATGCCG*, we mask the adapter sequences from raw reads in the file *test_NHFF.fastq* with the following command:

```
$ ./trim-adapter -o test_NHFF_adapter_masked.fastq
-a GATCGGAAGAGCGGTTCAGCAGGAATGCCGAGACCGATCTCGTATGCCG test_NHFF.fastq
```

If you would like to know the effective read length distribution after trimming adapter sequences and Ns from 3' end, you can add **-S** option to specify the output file for the distribution of effective read lengths. For example,

```
./trim-adapter -S test_NHFF_read_length_distr.txt -o test_NHFF_adapter_masked.fastq
-a GATCGGAAGAGCGGTTCAGCAGGAATGCCGAGACCGATCTCGTATGCCG test_NHFF.fastq
```

The file *test_NHFF_read_length_distr.txt* contains distribution of effective reads lengths after masking adapter sequences. Effective read length is the length of a read without Ns at the end of the read.

While we provide this **trim-adapter** as a standalone program for users who choose to map reads with another tool than rmapbs, its functionality is included in the **rmapbs** tool.

OPTIONS:

- **-a** <string>: adapter sequence
- **-o** <string>: output file (default: stdout)
- **-S** <string>: output file with effective read length distribution
- **-N** : for counting effective read length (without this option effective length includes the length of read without masked adapter sequence if any, and with this option it also includes the length of read without Ns at the end of the read)

INPUT: reads in FASTQ format.
OUTPUT: reads in FASTQ format.

1.2 Assessing read quality

It is optional to assess read quality before mapping, however such assessment may help us to spot potential problems during library preparation and/or bisulfite sequencing. The program **read-quality-prof** is used to generate an average summary of base composition and quality scores from 5' to 3' along all reads. We run **read-quality-prof** as following

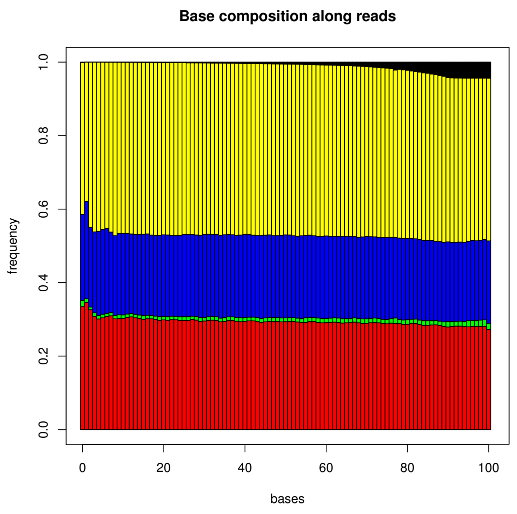
```
$ ./read-quality-prof -o test_NHFF_qual.txt test_NHFF.fastq
```

The output file *test_NHFF_qual.txt* can be visualized with the R software. Fig. **??** shows the base composition profile in the pair-end sequencing sample, where the left panel shows T-rich reads and the right panel A-rich reads. Note the small proportion of C and G reflects the effect of bisulfite conversion.

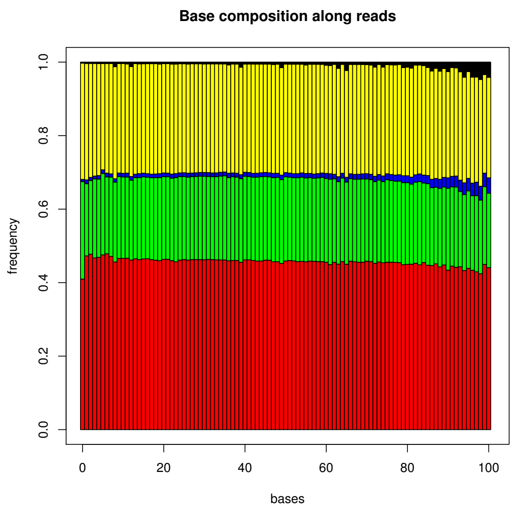
OPTIONS:

- **-o** <string>: output file (default: stdout)

INPUT: reads in FASTQ format.
OUTPUT: self-explanatory column names are provided in the output file



(a) T-rich reads



(b) A-rich reads

Figure 2: Base composition

2 Mapping

In the mapping step, you will map sequence reads to a reference genome. During bisulfite treatment, unmethylated cytosines in the original DNA sequences are converted to uracils, which are in turn incorporated as thymines during PCR amplification. We call such strand T-rich and its complementary strand A-rich (adenine-rich). When mapping T-rich reads to the reference genome, either a cytosine (C) or a thymine (T) in reads is considered valid match to a cytosine in the reference genome. But for A-rich reads, a adenine or a guanine is consider valid match to a guanine in the reference genome.

If using single-end sequencing, you will get T-rich reads only. If using pair-end sequencing, you will get both T-rich reads and A-rich reads. Next we will learn how to map bisulfite treated reads with **rmapbs** from either single-end sequencing or pair-end sequencing.

2.1 Mapping single-end sequencing reads

Mapping read sequences to the reference genome is done by **rmapbs**. Before mapping, you need to get he genome sequences of appropriate organism and assembly, which is usually downloadable from UCSC Genome Browser download portal or other specific repositories of genome sequences, such as TAIR for Arabidopsis *thaliana*. Suppose that you have already downloaded the appropriate genome sequences, in our example human genome hg18, in the directory *human-genome*, to map the reads in bcell/s-1.fq to the reference genome we run

```
$ ./rmapbs -c human-genome -s fa -m 4 -o bcell/s-1.bed bcell/s-1.fq -v.
```

The option **-c** specifies the direcotory holding the reference genome and the **-s** specifies the suffix of sequence file names. The option **-m** gives the maximum number of mismatches allowed. The option **-o** specifies the output file. There are other options to fine tune the behavior of **rmapbs** which are explained on Methpipe website.

2.2 Mapping pair-end sequencing reads

At present our pipeline map R-rich reads and A-rich reasd separately when mapping reads from the pair-end sequencing procedure. We map T-rich reads in the same way as for single end-reads. However we use A/G wildcards when mapping the A-rich reads, i.e. a adenine or a guanine is considered valid match to a guanine in the reference genome. The **rmapbs**, given the option **-A**, run in A/G wildcard mode. Therefore to map T-rich reads, for example, neut/s-1-1.fq, we run

```
$ ./rmapbs -c human-genome -s fa -m 4 -o neut/s-1-1.bed neut/s-1-1.fq -v.
```

To map A-rich reads, for example neut/s-1-2.fq, we run

```
$ ./rmapbs -c human-genome -s fa -m 4 -o neut/s-1-2.bed neut/s-1-2.fq -v -A.
```

The other options have the same meaning in both C/T wildcard mode and A/G wildcard mode.

2.3 Parallel mapping

Mapping is the most time-consuming and resource-demanding step in the Methpipe pipeline. Fortunately we may adopt a "divide and conquer" strategy to speed up this step. In brief, we first divide large sequence files into smaller ones, map them separately in parallel and finally combining the mapped files.

A large sequence file is divided into smaller ones with the **split** utility which is a standard utility in any UNIX-like platforms. By running

```
$ split -l 16000000 bcell/s-1.fq s-1
```

you will get a series of smaller files: s-1aa, s-1ab, s-1ac, etc, which can be mapped with **rmapbs** separately as described above. To learn about the detailed usage of **split**, you may read its man page. In brief, the **-l** option specifies the number of lines in each smaller file. Since FASTQ format requires that each read consist of four lines, i.e. name line, sequence line, name line and score line, the number of lines should be multipliers of 4. The optimal number depends on the time constraints and memory contraints of your computing platform. For your information, mapping 28 millions reads to the Arabidopsis *thaliana* genome with **rmapbs** uses about 40 minutes and 5Gb memory.

3 Post-mapping processing

If everything goes well, you should already have pre-processed raw sequence reads and mapped them to the reference genome. Suppose the mapped reads files of B cell sample are

```
bcell/s-1.bed bcell/s-2.bed bcell/s-3.bed bcell/s-4.bed,
```

and the mapped reads files of the neutrophile sample methylome are

```
neut/s-1-1.bed neut/s-1-2.bed neut/s-2-1.bed neut/s-2-2.bed.
```

We will discuss some necessary processing steps, including converting A-rich reads to T-rich reads in pair-end sequencing, masking overlapping regions of paired reads, handling duplicate reads and utilities to sort and/or merge mapped reads.

3.1 Converting A-rich reads to T-rich reads

As has been noted before, if using pair-end sequencing, you will get both T-rich reads and A-rich reads. When estimating methylation frequency of a cytosine residue, we count the number of cytosines, which originate from methylated cytosines, and the number of thymines, which originate from unmethylated cytosines, and the use the ratio $\frac{\#C}{\#C+\#T}$ as the estimate of methylation frequency of that base. Therefore, A-rich reads need to be converted to T-rich reads before we are able to leverage the information from these A-rich reads. This conversion is easily done with the program **revcomp**, for example

```
$ ./revcomp < neut/s-2-2.bed > neut/s-2-2-revcomp.bed
```

Note this conversion applies to and only applies to A-rich reads in pair-end sequencing. For notation convenience, we again assume you rename the converted reads file name to its original file name thereafter.

3.2 Joining paired reads

With pair-end sequencing, we obtain two reads: a T-rich read from 5' end of the orignal segments and an A-rich read from 5' end. After converting A-rich reads to T-rich reads, we have two T-rich reads from the orignal DNA segment, which possibly overlap for certain proportion. This overlapped region, if left as-is, will result in bias that cytosines (or thymines if unmethylated) from the same DNA molecule are counted twice.

The program **clipmates** identifies overlapped regions between two mates, join these two mates and output the DNA segment from which these two mates are originated. In our example project, neut/s-1-1.bed and neut/s-1-2.bed contain T-rich reads and A-rich reads (assume already converted to T-rich) respectively. We first sort the reads in these two files by name by running

```
$ sort -k4,4 neut/s-1-1.bed -o tmpfile && mv tmpfile neut/s-1-1.bed
$ sort -k4,4 neut/s-1-2.bed -o tmpfile && mv tmpfile neut/s-1-2.bed,
```

And then we run **clipmates** to join the T-rich mate and A-rich mate,

```
$ ./clipmates neut/s-1-1.bed neut/s-1-2.bed -o neut/s-1.bed
```

Note the input files to **clipmates**, input-file-1 and input-file-2, should strictly follow the order of T-rich and A-rich. The output file, specified by the option **-o**, contains the DNA segments after joining paired T-rich and A-rich mates. If two mates are mismatched (such as on different chromosomes or strands) or one mate is missing, **clipmates** simply outputs the original mates.

The program **clipmates** also provide statistical information about the pair-end sequencing data, such as the DNA fragment length distribution and the number of unpaired reads after mapping. This is done by specifying the fragment length file with the **-f** option. For example,

```
$ ./clipmates -f neut/s-1-len.txt -o neut/s-1.bed neut/s-1-1.bed neut/s-1-2.bed
```

In the file neut/s-1-len.txt **clipmates** writes the fragment length based on each pair of reads and marked reads that are unpaired. This file can be analyzed with standard UNIX text-processing utilities and statistical software.

3.3 Sorting reads

During the post-mapping processign of mapped reads, we may need to sort reads according to different criterions. This is done with the **sort** utility provided in UNIX environment.

In our pipeline, we store mapped reads in an extended BED format. The first six column are the same as the standard six-column BED format, which gives the chromosome name, starting position, ending position, read name, mismatches and strand. The seventh column gives the sequence of that read and the eighth reads the Solexa quality score. In order to sort reads according to their genomic locations, we run

```
$ sort -k1,1 -k2,3n -k6,6 bcell/s-1.bed > output-file.
```

To sort reads according to their names, we run

```
$ sort -k4,4 bcell/s-1.bed > output-file.
```

3.4 Removing duplicate reads

Because of PCR bias and other unknown causes, there are tens of thousands of reads mapped to exactly the same location in some regions of the genome while the average sequencing depth is usually below 20. These large amounts of reads are likely orginated from the same DNA segments and will bias the estimation of methylation frequency. One approach to alleviate this bias is to randomly pick one read from multiple reads that are generated the same genomic position. This task is done by the program unique **noduplicates**. For example if we would like to remove duplicate reads from the file bcell/s-1.bed, we run

```
$ ./noduplicates bcell/s-1.bed -o bcell/s-1-unique.bed
```

3.5 Merging read files

The final step before estimating methylation frequency and performing other higher level analysis is to merge read files from different sequencing lanes and/or flowcells.

If all read files are generated from the same biological library, we first sort all read files by genomic locations with **sort** (Section **??**), merge those files with **mergelanes** and then remove duplicate reads with **noduplicates** (Section **??**). We show how to merge read files in the B cell data below.

```
$ # sort read files
$ sort -k1,1 -k2,3n -k6,6 bcell/s-1.bed > tmpfile && mv tmpfile bcell/s-1.bed
$ sort -k1,1 -k2,3n -k6,6 bcell/s-2.bed > tmpfile && mv tmpfile bcell/s-2.bed
$ sort -k1,1 -k2,3n -k6,6 bcell/s-3.bed > tmpfile && mv tmpfile bcell/s-3.bed
$ sort -k1,1 -k2,3n -k6,6 bcell/s-4.bed > tmpfile && mv tmpfile bcell/s-4.bed
$ ./mergelanes bcell/s-*bed |./noduplicates > bcell/bcell.bed
```

If reads are generated from different biological libraries, it takes some extra step. The reason is that: even if two reads from different biological libraries are mapped to the same location, they are not originated from the same DNA molecule, therefore both should be kept for downstream analysis. Therefore, for reads from the same biological library, we pool reads and remove duplicates as described above. Then we just pool reads from different biological libraries without removing duplicates. This is controlled by the **-D** option of **mergelanes**. By default, **mergelanes** removes duplicate reads. If we would like to keep duplicate reads from different libraries, we specify this **-D** option. For example, we want to combine reads in *s-1-lib1.bed* from the first library and reads in *s-1-lib2.bed* from the second library, we run

```
$ ./mergelanes -D s-1-lib1.bed s-1-lib2.bed -o s-1.bed,
```

The file *s-1.bed* contains reads from the two libraries.

4 Methylation frequency and bisulfite conversion rate

At this step, we have precessed mapped reads and combines reads from different lanes and/or flowcells into a single file, which means that we have a bcell/bcell.bed containing all reads from the B cell sample and a neut/neut.bed containing all reads from the neutrophile sample. From now on, we will be working with these two aggregated files. In this section and the following section, we will proceed from analyzing methylation status at single loci to identifying higher methylation patterns such as hypo-methylated regions.

4.1 Estimating methylation frequency

In this section we will learn how to estimate the methylation probability at a single cytosine loci. In mammals, DNA methylation exists mostly at cytosines in the context of CpG dinucleotides. This kind of methylation is symmetric because the cytosine on the complementary strand, which is base-paired with the guanine, is also methylated due to the interesting property of maintenance DNA methyltransferases. In mammalian stem cells and particularly in plant cells, cytosines in sequence contexts other than CpG dinucleotides, such as CpHpG (H denotes adenines, thymines or cytosines), may also be methylated. This type of methylation is called asymmetric as the cytosines on the complementary strand are unnecessarily methylated. Symmetric methylation and asymmetric methylation requires different methods to estimate the methylation frequency. Simply put, in the case of asymmetric methylation we can use only reads mapped to one strand while we can use reads mapped to both strands in the case of symmetric methylation. Later we will describe the estimation methods for these two cases separately, first the case of asymmetric methylation and then the case of symmetric methylation.

4.1.1 Methylation level in non-CpG cytosines

After bisulfite sodium treatment, unmethylated cytosines in the original DNA sequences are converted to thymines while methylated cytosines remain the same. For convenience, we assume the bisulfite conversion is perfect for now.

For a cytosine in the non-CpG context, we count the number of cytosines, which originate from methylated cytosines, and the number of thymines, which originate from unmethylated cytosines, from reads in the same strand as the cytosine. The unbiased estimator of the methylation frequency at this cytosine is simply the ratio of the number of cytosines to the number of cytosines and thymines combined. This task is carried out with the program **methcount** as following:

```
$ ./methcounts -c human_genome -non -o bcell/bcell.bed bcell/bcell-methcounts.bed,
```

where the **-c** options specifies the direcotory containing the human genome sequences and the **-non** options indicates the program should run in non-CpG mode. For each cytosine, the program reports the number of cytosines, the number of thymines and the estimation of methylation frequency in the output file bcell/bcell-methcounts.bed.

4.1.2 Methylation level in CpG cytosines

For cytosines in the context of CpG dinucleotides, we are able to use reads mapped to both strands, which means better coverage and therefore reduces variance. We count the number of cytosines and thymines in all reads that overlap cytosines either in the positive strand or in the negative strand. And the estimator of methylation frequency is still the ratio of the number of cytosines to the number of cytosines and thymines combined. This task is done simiarly to the case of non-CpG methylation except that we ignore the **-non** option of the **methcount** program, for example,

```
$ ./methcounts -c human_genome -o bcell/bcell-methcounts.bed bcell/bcell.bed.
```

The methcount program report the methylation status of only those cytosines in the CpG context.

4.2 Estimating bisulfite conversion rate

By bisulfite sodiom treatment, unmethylated cytosines are converted to uracils, which are later read out as thymines. However this conversion may be incomplete due to various reasons, such as insufficient concentration, insufficient time of treatment or sequencing error. Therefore, bisulfite conversion rate, defined as ratio of converted unmethylated cytosines to all unmethylated cytosines, is an important parameter to assess the data quality of a BS-Seq experiment. To stimate bisulfite conversion rate, we need first to have some cytosines in the genome that are unmethylated. This includes three scenarios: (i) non-CpG cytosines in most mammalian tissues, (ii) spike-in Lamda genome and (iii) the chloroplast genome in plants. In these three scenarios, unmethylated cytosines should all be converted if the bisulfite treatment is perfect. Otherwise, some cytosines are left as-is. Therefore we count the number of cytosines covering these unmethylated cytosines and the number of thymines, the ratio of thymines to cytosines and thymines combined is the estimator of the bisulfite conversion rate.

Estimating bisulfite conversion rate from non-CpG cytosines in mammals: In most mammalian tissues, non-CpG cytosines are free from methylation (though significant proportion of non-CpG methylation is observed in human stem cells), therefore we assume non-CpG cytosines are unmethylated and use the **bsrate** program to estimate conversion rate as following:

```
$ ./bsrate -c human_genome -o bcell/conversion-rate.txt bcell/bcell.bed.
```

The output file *conversion-rate.txt* gives the conversion rate according to strands and locations.

Estimating bisulfite conversion rate from spike-in Lambda genome: The genome of Lambda phage is free from methylation. Therefore in some experiment setting, certain amount of Lambda phage genome are added to the biological sample of interest, subject to the same bisulfite treatment and sequencing. Suppose that the DNA sequences for Lambda phage are stored in the directory *lambda-genome* and the reads mapped to the Lambda genome are stored in the file *lambda.bed*, we issue the **bsrate** program as following:

```
$ ./bsrate -c lambda-genome -o conversion-rate.txt -A lambda.bed.
```

Note the **-A** option: it tells **bsrate** to use information of all cytosines since all cytosines in the Lambda genome, no matter their sequence context, are unmethylated.

Estimating bisulfite conversion rate from chloroplast genome in plants: If you are working with plants with chloroplast genome, we can also use cytosines there to estimate bisulfite conversion rate because chloroplast genome are unmethylated. The command is similar with above, i.e.,

```
$ ./bsrate -c plant-genome -o conversion-rate.txt -A plan-chloroplast.bed.
```

5 Higher level analysis

5.1 Hypo-methylated regions

5.2 Differentially methylated regions

5.3 Allelic-specifically methylated regions

6 Summary and Visualization

7 A sample work flow

This part shows how these tools are connected to get the methylation profile. Suppose we have the following BS-Seq library

```
reads/s_1_1.txt reads/s_1_2.txt reads/s_2_1.txt reads/s_2_2.txt
```

The final result can be obtained as following. For clarity, we show all the intermediate result. In real application, some intermediate files can be avoided by using pipes.

```
# Pre-mapping processing #

## trim adapter ##
$ ./trim-adapter reads/s_1_1.txt preprocessed/s_1_1.txt
$ ./trim-adapter reads/s_1_2.txt preprocessed/s_1_2.txt
$ ./trim-adapter reads/s_2_1.txt preprocessed/s_2_1.txt
$ ./trim-adapter reads/s_2_2.txt preprocessed/s_2_2.txt

# Mapping #
$ ./rmapbs -c genome_seq_dir -o mapped/s_1_1.mr preprocessed/s_1_1.txt
$ ./rmapbs -A -c genome_seq_dir -o mapped/s_1_2.mr preprocessed/s_1_2.txt
$ ./rmapbs -c genome_seq_dir -o mapped/s_2_1.mr preprocessed/s_2_1.txt
$ ./rmapbs -A -c genome_seq_dir -o mapped/s_2_2.mr preprocessed/s_2_2.txt

# post-mapping processing

## reverse complement A-rich strand ##
$ ./revcomp mapped/s_1_2.mr > tmpfile && mv tmpfile mapped/s_1_2.mr
$ ./revcomp mapped/s_2_2.mr > tmpfile && mv tmpfile mapped/s_2_2.mr

## mask overlapping ##
#### first sort by name ####
$ ./sort -N mapped/s_1_1.mr -o tmpfile && mv tmpfile  mapped/s_1_1.mr
$ ./sort -N mapped/s_1_2.mr -o tmpfile && mv tmpfile  mapped/s_1_2.mr
$ ./sort -N mapped/s_2_1.mr -o tmpfile && mv tmpfile  mapped/s_2_1.mr
$ ./sort -N mapped/s_2_2.mr -o tmpfile && mv tmpfile  mapped/s_2_2.mr
```

```
#### masking ####
$ ./mask-overlap mapped/s_1_1.mr mapped/s_1_2.mr masked/s_1_1.mr masked/s_1_2.mr
$ ./mask-overlap mapped/s_2_1.mr mapped/s_2_2.mr masked/s_2_1.mr masked/s_2_2.mr

#### sort by genomic location ####
$ ./sort masked/s_1_1.mr -o tmpfile && mv tmpfile  masked/s_1_1.mr
$ ./sort masked/s_1_2.mr -o tmpfile && mv tmpfile  masked/s_1_2.mr
$ ./sort masked/s_2_1.mr -o tmpfile && mv tmpfile  masked/s_2_1.mr
$ ./sort masked/s_2_2.mr -o tmpfile && mv tmpfile  masked/s_2_2.mr

## combine all result ##
#### merge ####
$ ./merge -o all.mr masked/s_1_1.mr masked/s_1_2.mr masked/s_2_1.mr masked/s_2_2.mr

#### jackpot removal #####
$ ./unique all.mr -o tmpfile && mv tmpfile all.mr

# analysis #
## methcounts ##
$ ./methcounts -c genome_sequence_file -o all-methcounts.bed all.mr
```