The Smithlab DNA Methylation Data Analysis Pipeline (MethPipe)

Qiang Song Benjamin Decato Michael Kessler Fang Fang Jenny Qu
Tyler Garvin Meng Zhou Andrew Smith

August 8, 2017

Contents

The `methpipe` software package is a comprehensive pipeline and set of tools for analyzing whole genome bisulfite sequencing data (WGBS). This manual explains the stages in our pipeline, how to use the analysis tools, and how to modify the pipeline for your specific context.

1 Assumptions

Our pipeline was designed to run in a cluster computing context, with many processing nodes available, and a job submission system like PBS or SGE. Much of this analysis is computationally intensive. We assume that individual nodes will have several GB of memory available for processing. Typically the data we deal with amounts to a minimum of 100GB for a mammalian methylome at 10x coverage. Intermediate files may cause this amount to more than double during execution of the pipeline, and likely at the end of the pipeline the total size of files will amount to almost double the size of the raw data.

Users are assumed to be quite familiar with UNIX/Linux and related concepts (*e.g.* building software from source, using the command line, shell environment variables, etc.).

It is also critical that users are familiar with bisulfite sequencing experiments, especially the bisulfite conversion reaction, and how this affects what we observe in the sequenced reads. This is especially important if paired-end sequencing is used. If you do not understand these concepts, you will likely run into major problems trying to customize our pipeline.

2 Methylome construction

2.1 Mapping reads

During bisulfite treatment, unmethylated cytosines in the original DNA sequences are converted to uracils, which are then incorporated as thymines (T) during PCR amplification. These PCR products are referred to as T-rich sequences as a result of their high thymine constitution. With paired-end sequencing experiments, the compliments of these T-rich sequences are also sequenced. These complimentary sequences have high adenine (A) constitution (A is the complimentary base pair of T), and are referred to as A-rich sequences. Mapping consists of finding sequence similarity, based on context specific criteria, between these short sequences, or reads, and an orthologous reference genome. When mapping T-rich reads to the reference genome, either a cytosine (C) or a thymine (T) in a read is considered a valid match for a cytosine in the reference genome. For A-rich reads, an adenine or a guanine is considered a valid match for a guanine in the reference genome. The mapping of reads to the reference genome by `WALT` is described below. If you choose to map reads with a different tool, make sure that your post-mapping files are appropriately formatted for the next components of the `methpipe` pipeline (necessary file formats for each step are covered in the corresponding sections). The default behavior of `WALT` is to assume that reads are T-rich and map accordingly. `WALT` is available on github at http://github.com/smithlabcode/walt.

Input and output file formats: We assume that the original data is a set of sequenced read files, typically as produced by Illumina sequencing. These are FASTQ format files, and can be quite large. After the reads are mapped, these files are not used by our pipeline. The reference genome should be a folder containing an individual FASTA (named like `*.fa`) file for each chromosome to maximize memory efficiency.

The mapped reads files (`*.mr` suffix) that result from the previous steps should consist of eight columns of data. The first six columns are the traditional components of a BED file (chromosome, start, end, read name, number of mismatches, strand), while the last two columns consist of sequence and quality scores respectively. These mapped reads files will be the input files for the postprocessing quality control and analysis programs to follow, including `bsrate` and `methcounts`.

`WALT` operates by preprocessing the reference genome into a large index, where k-mers of set length are used as keys to a list of potential mapping locations for reads that begin with their sequence. Keeping in mind that this file may be quite large (40GB for mm10), to produce this index run the following command:

```
makedb -c <genome folder or file> -o <index file>
```

Decompressing and isolating paired-end reads: Sometimes paired-end reads are stored in the same FASTQ file. Because we treat these paired ends differently, they must be separated into two files and run through `WALT` with different parameters.

If your data is compressed as a Sequenced Read Archive, or SRA file, you can decompress and split paired-end reads into two files at the same time using `fastq-dump`, which is a program included in the `sra-toolkit` package, available for most unix systems. Below is an example of using `fastq-dump` to decompress and separate FASTQ data by end:

```
$ fastq-dump --split-3 Human_ESC.sra
```

If you have a FASTQ file not compressed in SRA format, you can split paired ends into two separate files by running the following commands:

```
$ sed -ne '1~8{N;N;N;p}' *.fastq > *_1.fastq
$ sed -ne '4~8{N;N;N;p}' *.fastq > *_2.fastq
```

Sequencing adapters: These are a problem in any sequencing experiment with short fragments relative to the lengths of reads. `WALT` identifies sequences at the ends of reads greater than 10bp belonging to sequencing adapters and converts them to Ns to avoid potential mapping problems.

Adapter sequences must be supplied to `WALT` through the `-C` option. Keep in mind that if the adapter sequence provided to you for the second paired end is displayed from 5' to 3', you will need to provide the reverse complement of the sequence to `WALT`.

A more robust method for removing adapters is available via the Babraham Institute's Bioinformatics group, called `trimgalore`. This program takes into account adapter contamination of fewer than 10 nucleotides and can handle mismatches with the provided sequence.

Single-end reads: When working with data from a single-end sequencing experiment, you will have T-rich reads only. `WALT` expects T-rich reads as a default. Execute the following command to map all of your single-end reads with `WALT`:

```
$ walt -i <index> -r <reads> -o <output file> [options]
```

Paired-end reads: When working with data from a paired-end sequencing experiment, you will have T-rich and A-rich reads. T-rich reads are often kept in files labeled with an "_1" and A-rich reads are often kept in files labeled with an "_2". T-rich reads are sometimes referred to as $5'$ reads or mate 1 and A-rich reads are sometimes referred to $3'$ reads or mate 2. We assume that the T-rich file and the A-rich contain the same number of reads, and each pair of mates occupy the same lines in their respective files. We will follow this convention throughout the manual and strongly suggest that you do the same. Run the following command to map two reads files from a paired-end sequencing experiment:

```
$ walt -i <index> -1 <t-rich reads> -2 <a-rich reads> -o <output file> [options]
```

In brief, what happens internally in `WALT` is as follows. `WALT` finds candidate mapping locations for a T-rich mate with CG-wildcard mapping, and candidate mapping locations for the corresponding A-rich mate with AG-wildcard mapping. If two candidate mapping locations of the pair of mates are within certain distance in the same chromosome and strand and with correct orientation, the two mates are combined into a single read (after reverse complement of the A-rich mate), referred to as a fragment. The overlapping region between the two mates, if any, is included once, and the gap region between them, if any, is filled with Ns. The parameter `-L` to `WALT` indicates the maximum size of fragments to allow to be merged.

Here the fragment size is the sum of the read lengths at both ends, plus whatever distance is between them. So this is the length of the original molecule that was sequenced, excluding the sequencing adapters. It is possible for a given read pair that the molecule was shorter than twice the length of the reads, in which case the ends of the mates will overlap, and so in the merged fragment will only be included once. Also, it is possible that the entire molecule was shorter than the length of even one of the mates, in which case the merged fragment will be shorter than either of the read ends. If the two mates cannot be merged because they are mapped to different chromosomes or different strand, or they are far away from each other, `WALT` will output each mate individually if its mapping position is unambiguous.

WALT provides a statistical summary of its mapping job in the "mapstats" file, which includes the total number and proportion of reads mapped, how many paired end mates were mapped together, and the distribution of fragment lengths computed by the matched pairs.

Mapping reads in a large file: Mapping reads often takes a while, and mapping reads from WGBS takes even longer. It usually take quite a long time to map reads from a single large file with tens of millions of reads. If you have access to a cluster, one strategy is to launch multiple jobs, each working on a subset of reads simultaneously, and finally combine their output. WALT takes advantage of OpenMP to parallelize the process of mapping reads using the shared index.

If each node can only access its local storage, dividing the set of reads to into k equal sized smaller reads files, and mapping these all simultaneously on multiple nodes, will make the mapping finish about k times faster. The unix split command is good for dividing the reads into smaller parts. The following BASH commands will take a directory named reads containing Illumina sequenced reads files, and split them into files containing at most 3M reads:

```
$ mkdir reads_split
$ for i in reads/*.txt; do \
    split -a 3 -d -l 12000000 ${i} reads_split/$(basename $i); done
```

Notice that the number of lines per split file is 12M, since we want 3M reads, and there are 4 lines per read. If you split the reads like this, you will need to "unsplit" them after the mapping is done. This can be done using the cat command.

Alternative mappers: In addition to WALT described above, users may also wish to process raw reads using alternative mapping algorithms, including BSSeeker, which uses a three nucleotide alphabet strategy, and BSMAP, which allows for gaps during mapping. The program to-mr is used to convert the output from those mappers to the *.mr format used in our pipeline. To convert BSMAP mapped read file in .bam format, run

```
$ to-mr -o Human_NHFF.mr -m bsmap Human_NHFF.bam
```

where the option -m specifies that the original mapper is BSMAP. To obtain a list of alternative mappers supported by our converter, run to-mr without any options.

2.2 Merging libraries and removing duplicates

Before calculating methylation level, you should now remove read duplicates, or reads that were mapped to identical genomic locations. These reads are most likely the results of PCR over-amplication rather than true representations of distinct DNA molecules. The program duplicate-remover aims to remove such duplicates. It collects duplicate reads and/or fragments that have identical sequences and are mapped to the same genomic location (same chromosome, same start and end, and same strand), and chooses a random one to be the representative of the original DNA sequence.

duplicate-remover can take reads sorted by (chrom, start, end, strand). If the reads in the input file are not sorted, run the following sort command:

```
$ LC_ALL=C sort -k 1,1 -k 2,2n -k 3,3n -k 6,6 \
      -o Human_ESC.mr.sorted_start Human_ESC.mr
$ LC_ALL=C sort -k 1,1 -k 2,2n -k 3,3n -k 6,6 \
      -o Human_NHFF.mr.sorted_start Human_NHFF.mr
```

Next, execute the following command to remove duplicate reads:

```
$ duplicate-remover -S Human_ESC_dremove_stat.txt \
                      -o Human_ESC.mr.dremove Human_ESC.mr.sorted_start
$ duplicate-remover -S Human_NHFF_dremove_stat.txt \
                      -o Human_NHFF.mr.dremove Human_NHFF.mr.sorted_start
```

The duplicate-removal correction should be done on a per-library basis, i.e, one should pool all reads from multiple runs or lanes sequenced from the same library and remove duplicates. The reads from distinct libraries can be simply pooled without any correction as the reads from each library are originated from distinct DNA fragments.

2.3 Estimating bisulfite conversion rate

Unmethylated cytosines in DNA fragments are converted to uracils by sodium bisulfite treatment. As these fragments are amplified, the uracils are converted to thymines and so unmethylated Cs are ultimately read as Ts (barring error). Despite its high fidelity, bisulfite conversion of C to T does have some inherent failure rate, depending on the bisulfite kit used, reagent concentration, time of treatment, etc., and these factors may impact the success rate of the reaction. Therefore, the bisulfite conversion rate, defined as the rate at which unmethylated cytosines in the sample appear as Ts in the sequenced reads, should be measured and should be very high (*e.g.* > 0.99) for the experiment to be considered a success.

Measuring the bisulfite conversion rate this way requires some kind of control set of genomic cytosines not believed to be methylated. Three options are (1) to spike in some DNA known not to be methylated, such as a Lambda virus, (2) to use the human mitochondrial genome, which is known to be entirely unmethylated, or chloroplast genomes which are believed not to be methylated, or (3) to use non-CpG cytosines which are believed to be almost completely unmethylated in most mammalian cells. In general the procedure is to identify the positions in reads that correspond to these presumed unmethylated cytosines, then compute the ratio of T to (C + T) at these positions. If the bisulfite reaction was perfect, then this ratio should be very close to 1, and if there is no bisulfite treatment, then this ratio should be close to 0.

The program `bsrate` will estimate the bisulfite conversion rate in this way. Assuming method (3) from the above paragraph of measuring conversion rate at non-CpG cytosines in a mammalian methylome, the following command will estimate the conversion rate.

```
$ bsrate -c hg38 -o Human_ESC.bsrate Human_ESC.mr.dremove
$ bsrate -c hg38 -o Human_NHFF.bsrate Human_NHFF.mr.dremove
```

Note that we used the output of `duplicate-remover` to reduce any bias introduced by incomplete conversion on PCR duplicate reads. The `bsrate` program requires that the input be sorted so that reads mapping to the same chromosome are contiguous and the input should have duplicate reads already removed to reduce bias. The first several lines of the output might look like the following:

```
OVERALL CONVERSION RATE = 0.994141
POS CONVERSION RATE = 0.994166  832349
NEG CONVERSION RATE = 0.994116  825919
BASE PTOT  PCONV PRATE   NTOT  NCONV NRATE   BTHTOT BTHCONV BTHRATE ERR ALL    ERRRATE
1    8964  8813  0.9831  9024  8865  0.9823  17988  17678   0.9827  95  18083  0.0052
2    7394  7305  0.9879  7263  7183  0.9889  14657  14488   0.9884  100 14757  0.0067
3    8530  8442  0.9896  8323  8232  0.9890  16853  16674   0.9893  98  16951  0.0057
4    8884  8814  0.9921  8737  8664  0.9916  17621  17478   0.9918  76  17697  0.0042
5    8658  8596  0.9928  8872  8809  0.9929  17530  17405   0.9928  70  17600  0.0039
6    9280  9218  0.9933  9225  9177  0.9948  18505  18395   0.9940  59  18564  0.0031
7    9165  9117  0.9947  9043  8981  0.9931  18208  18098   0.9939  69  18277  0.0037
8    9323  9268  0.9941  9370  9314  0.9940  18693  18582   0.9940  55  18748  0.0029
9    9280  9228  0.9944  9192  9154  0.9958  18472  18382   0.9951  52  18524  0.0028
10   9193  9143  0.9945  9039  8979  0.9933  18232  18122   0.9939  66  18298  0.0036
```

The above example is based on a very small number of mapped reads in order to make the output fit the width of this page. The first thing to notice is that the conversion rate is computed separately for each strand. The information is presented separately because this is often a good way to see when some problem has occurred in the context of paired-end reads. If the conversion rate looks significantly different between the two strands, then we would go back and look for a mistake that has been made at an earlier stage in the pipeline. The first 3 lines in the output indicate the overall conversion rate, the conversion rate for positive strand mappers, and the conversion rate for negative strand mappers. The total number of nucleotides used (*e.g.* all C+T mapping over genomic non-CpG C's for method (3))

is given for positive and negative strand conversion rate computation, and if everything has worked up to this point these two numbers should be very similar. The 4th line gives column labels for a table showing conversion rate at each position in the reads. The labels PTOT, PCONV and PRATE give the total nucleotides used, the number converted, and the ratio of those two, for the positive-strand mappers. The corresponding numbers are also given for negative strand mappers (NTOT, NCONV, NRATE) and combined (BTH). The sequencing error rate is also shown for each position, though this is an underestimate because we assume at these genomic sites any read with either a C or a T contains no error.

The bisulfite conversion rate reported in MethBase is computed from all non-CpG cytosines, and assumes zero non-CpG methylation. Because this is likely not true, we report a conservative estimate of conversion rate. A better method for computing conversion rate is to use an unmethylated spike-in or reads mapping to mitochondria, which has been shown to be entirely unmethylated in most human tissues. To use the mitochondria, you can use the following command:

```
$ grep ^chrM Human_ESC.mr > Human_ESC.mr.chrM
$ bsrate -N -c chrM.fa -o Human_ESC.bsrate Human_ESC.mr.chrM
```

After completing bisulfite conversion rate analysis, remember to remove any control reads not naturally occurring in the sample (lambda virus, mitochondrial DNA from another organism, etc.) before continuing. The output from two different runs of `bsrate` can be merged using the program `merge-bsrate`.

2.4 Computing single-site methylation levels

The `methcounts` program takes the mapped reads and produces the methylation level at each genomic cytosine, with the option to produce only levels for CpG-context cytosines. While most DNA methylation exists in the CpG context, cytosines in other sequence contexts, such as CXG or CHH (where H denotes adenines, thymines, or cytosines and X denotes adenines or thymines) may also be methylated. Non-CpG methylation occurs most frequently in plant genomes and pluripotent mammalian cells such as embryonic stem cells. This type of methylation is asymmetric since the cytosines on the complementary strand do not necessarily have the same methylation status.

The input is in mapped read format. The reads should be sorted and duplicate reads should be removed (same steps as in section 2.2). If your reads are not sorted, run:

```
$ LC_ALL=C sort -k 1,1 -k 2,2n -k 3,3n -k 6,6 \
       -o Human_ESC.mr.sorted_start Human_ESC.mr
```

If duplicate reads are not removed, run:

```
$ duplicate-remover -S Human_ESC_dremove_stat.txt \
                      -o Human_ESC.mr.dremove Human_ESC.mr.sorted_start
```

Running `methcounts`: The methylation level for every cytosine site at single base resolution is estimated as a probability based on the ratio of methylated to total reads mapped to that loci. Because not all DNA methylation contexts are symmetric, methylation levels are produced for both strands and can be analyzed separately. To compute methylation levels at each cytosine site along the genome you can use the following command:

```
$ methcounts -c hg38 -o Human_ESC.meth \
               Human_ESC.mr.sorted_start
$ methcounts -c hg18 -o Human_NHFF.meth \
               Human_NHFF.mr.sorted_start
```

The argument `-c` gives the filename of the genome sequence or the directory that contains one FASTA format file for each chromosome. By default `methcounts` identifies these chromosome files by the extension `.fa`. Importantly, the "name" line in each chromosome file must be the character > followed by the same name that identifies that chromosome in the mapped read output (the `.mr` files). An example of the output and explanation of each column follows:

```
chr1 3000826 + CpG 0.852941 34
chr1 3001006 + CHH 0.681818 44
chr1 3001017 -CpG 0.609756 41
chr1 3001276 + CpGx 0.454545 22
chr1 3001628 -CHH 0.419753 81
chr1 3003225 + CpG 0.357143 14
chr1 3003338 + CpG 0.673913 46
chr1 3003378 + CpG 0.555556 27
chr1 3003581 -CHG 0.641026 39
chr1 3003639 + CpG 0.285714 7
```

The output file contains one line per cytosine site. The first column is the chromosome. The second is the location of the cytosine. The 3rd column indicates the strand, which can be either + or −. The 4th column is the sequence context of that site, followed by an x if the site has mutated in the sample away from the reference genome. The 5th column is the estimated methylation level, equal to the number of Cs in reads at position corresponding to the site, divided by the sum of the Cs and Ts mapping to that position. The final column is number of reads overlapping with that site.

Note that because `methcounts` produces a file containing one line for every cytosine in the genome, the file can get quite large. For reference assembly mm10, the output is approximately 25GB. The `-n` option produces methylation data for CpG context sites only, and for mm10 this produces an output file that is approximately 1GB. It is recommended that users allocate at least 8GB of memory when running `methcounts`.

To examine the methylation status of cytosines a particular sequence context, one may use the `grep` command to filter those lines based on the fourth column. For example, in order to pull out all cytosines within the CHG context, run the following:

```
$ grep CHG Human_ESC_All.meth > Human_ESC_CHG.meth
```

Our convention is to name `methcounts` output with all cytosines like `*_All.meth`, with CHG like `*_CHG.meth` and with CHH like `*_CHH.meth`.

Extracting and merging symmetric CpG methylation levels: Since symmetric methylation level is the common case for CpG methylation, we have designed all of our analysis tools based on symmetric CpG sites, which means each CpG pair generated by `methcounts` should be merged to one. The `symmetric-cpgs` program is used to merge those symmetric CpG pairs. It works for `methcounts` output with either all cytosines or CpGs only (generated with `-n` option).

```
$ symmetric-cpgs -o Human_ESC_CpG.meth Human_ESC_ALL.meth
```

The above command will merge all CpG pairs while throwing out mutated sites. Note that as long as one site of the pair is mutated, the whole pair will be discarded. This default mode is recommended. If one wants to keep those mutated pairs, run

```
$ symmetric-cpgs -m -o Human_ESC_CpG.meth Human_ESC_ALL.meth
```

Merging methcounts files from multiple replicates: When working with a BS-seq project with multiple replicates, you may first produce a methcounts output file for each replicate individually and assess the reproducibility of the methylation result by comparing different replicates. The `merge-methcounts` program is used to merge the those individual methcounts file to produce a single estimate that has higher coverage. Suppose you have the three methcounts files from three different biological replicates, `Human_ESC_R1/R1.meth`, `Human_ESC_R2/R2.meth` and `Human_ESC_R3/R3.meth`. To merge those individual methcounts files, execute

```
$ merge-methcounts Human_ESC_R1/R1.meth  Human_ESC_R2/R2.meth \
  Human_ESC_R3/R3.meth -o Human_ESC.meth
```

Computation of methylation level statistics The `levels` program computes statistics for the output of `methcounts`. Sample output is below. It computes the total fraction of cytosines covered, the fraction of cytosines that have mutated away from the reference, and coverage statistics for both CpGs and all cytosines. For CpG sites, coverage number reflects taking advantage of their symmetric nature and merging the coverage on both strands. For CpG coverage minus mutations, we remove the reads from CpG sites deemed to be mutated away from the reference. It also computes average methylation in three different ways, described in Schultz et al. (2012). This program should provide flexibility to compare methylation data with publications that calculate averages different ways and illustrate the variability of the statistic depending on how it is calculated.

```
SITES: 1000000
SITES COVERED: 566157
FRACTION MUTATED: 0.001257
FRACTION COVERED: 0.566157
MAX COVERAGE: 439
SYMMETRICAL CpG COVERAGE: 11.3228
SYMMETRICAL CpG COVERAGE (WHEN > 0): 15.3289
SYMMETRICAL CpG COVERAGE (minus mutations): 11.2842
SYMMETRICAL CpG COVERAGE (WHEN > 0) (minus mutations): 15.2768
MEAN COVERAGE: 3.31458
MEAN COVERAGE (WHEN > 0): 5.85452
METHYLATION LEVELS (CpG CONTEXT):
mean_meth 0.700166
w_mean_meth 0.667227
frac_meth 0.766211
METHYLATION LEVELS (CHH CONTEXT):
mean_meth 0.0275823
w_mean_meth 0.0184198
frac_meth 0.0146346
METHYLATION LEVELS (CXG CONTEXT):
mean_meth 0.0217537
w_mean_meth 0.0170535
frac_meth 0.00843068
METHYLATION LEVELS (CCG CONTEXT):
mean_meth 0.0211243
w_mean_meth 0.0187259
frac_meth 0.00630109
```

To run the `levels` program, execute

```
$ levels -o Human_ESC.levels Human_ESC.meth
```

3 Methylome analysis

The following tools will analyze much of the information about CpGs generated in previous steps and produce methylome wide profiles of various methylation characteristics. In the context of Methpipe, these characteristics consist of hypomethylated regions (HMRs), partially methylated regions (PMRs), differentially methylated regions between two methylomes (DMRs), regions with allele-specific methylation (AMRs), and hydroxymethylation.

3.1 Hypomethylated and hypermethylated regions

The distribution of methylation levels at individual sites in a methylome (either CpGs or non-CpG Cs) almost always has a bimodal distribution with one peak low (very close to 0) and another peak high (close to 1). In most mammalian

cells, the majority of the genome has high methylation, and regions of low methylation are typically more interesting. These are called *hypo-methylated regions* (HMRs). In plants, most of the genome has low methylation, and it is the high parts that are interesting. These are called *hyper-methylated regions*. For stupid historical reasons in the Smith lab, we call both of these kinds of regions HMRs. One of the most important analysis tasks is identifying the HMRs, and we use the `hmr` program for this. The `hmr` program uses a hidden Markov model (HMM) approach using a Beta-Binomial distribution to describe methylation levels at individual sites while accounting for the number of reads informing those levels. `hmr` automatically learns the average methylation levels inside and outside the HMRs, and also the average size of those HMRs.

Requirements on the data: We typically like to have about 10x coverage to feel very confident in the HMRs called in mammalian genomes, but the method will work with lower coverage. The difference is that the boundaries of HMRs will be less accurate at lower coverage, but overall most of the HMRs will probably be in the right places if you have coverage of 5-8x (depending on the methylome). Boundaries of these regions are totally ignored by analysis methods based on smoothing or using fixed-width windows.

Typical mammalian methylomes: Running `hmr` requires a file of methylation levels formatted like the output of the `methcounts` program (as described in Section **??**). For calling HMRs in mammalian methylomes, we suggest only considering the methylation level at CpG sites, as the level of non-CpG methylation is not usually more than a few percent. The required information can be extracted and processed by using `symmetric-cpgs` (see Section **??**).

```
$ symmetric-cpgs -o Human_ESC_CpG.meth Human_ESC_ALL.meth
$ hmr -o Human_ESC.hmr Human_ESC.meth
```

The output will be in BED format, and the indicated strand (always positive) is not informative. The name column in the output will just assign a unique name to each HMR, and the score column indicates how many CpGs exist inside the HMR. Each time the `hmr` is run it requires parameters for the HMM to use in identifying the HMRs. We usually train these HMM parameters on the data being analyzed, since the parameters depend on the average methylation level and variance of methylation level; the variance observed can also depend on the coverage. However, in some cases it might be desirable to use the parameters trained on one data set to find HMRs in another. The option `-p` indicates a file in which the trained parameters are written, and the argument `-P` indicates a file containing parameters (as produced with the `-p` option on a previous run) to use:

```
$ hmr -p Human_ESC.hmr.params -o Human_ESC.hmr Human_ESC.meth
$ hmr -P Human_ESC.hmr.params -o Human_NHFF_ESC_params.hmr Human_NHFF.meth
```

In the above example, the parameters were trained on the ESC methylome, stored in the file `Human_ESC.hmr.params` and then used to find HMRs in the NHFF methylome. This is useful if a particular methylome seems to have very strange methylation levels through much of the genome, and the HMRs would be more comparable with those from some other methylome if the model were not trained on that strange methylome.

Plant (and similar) methylomes: The plant genomes, exemplified by *A. thaliana*, are devoid of DNA methylation by default, with genic regions and transposons being hyper-methylated, which we termed HyperMRs to stress their difference from *hypo-methylated regions* in mammalian methylomes. DNA methylation in plants has been associated with expression regulation and transposon repression, and therefore characterizing HyperMRs is of much biological relevance. In addition to plants, hydroxymethylation tends to appear in a small fraction of the mammalian genome, and therefore it makes sense to identify hyper-hydroxymethylated regions.

The first kind of HyperMR analysis involves finding continuous blocks of hyper-methylated CpGs with the `hmr` program. Since `hmr` is designed to find hypo-methylated regions, one needs first to invert the methylation levels in the `methcounts` output file as follows:

```
$ awk '{$5=1-$5; print $0}' Col0.meth > Col0_inverted.meth
```

Next one may use the `hmr` program to find "valleys" in the inverted Arabidopsis methylome, which are the hyper-methylated regions in the original methylome. The command is invoked as below

```
$ hmr -o Col0.hmr Col0_inverted.meth
```

This kind of HyperMR analysis produces continuous blocks of hyper-methylated CpGs. However in some regions, intragenic regions in particular, such continuous blocks of hyper-methylated CpGs are separated by a few unmethylated CpGs, which have distinct sequence preference when compared to those CpGs in the majority of unmethylated genome. The blocks of hyper-methylated CpGs and gap CpGs together form composite HyperMRs. The `hypermr` program, which implements a three-state HMM, is used to identify such HyperMRs. Suppose the `methcounts` output file is `Col0_Meth.bed`, to find HyperMRs from this dataset, run

```
$ hypermr -o Col0.hypermr Col0.meth
```

The output file is a 6-column BED file. The first three columns give the chromosome, starting position and ending position of that HyperMR. The fourth column starts with the "hyper:", followed by the number of CpGs within this HyperMR. The fifth column is the accumulative methylation level of all CpGs. The last column indicates the strand, which is always +.

Lastly, it is worth noting that plants exhibit significantly more methylation in the non-CpG context, and therefore inclusion of non-CpG methylation in the calling of hyper-methylated regions could possibly be informative. We suggest separating each cytosine context from the `methcounts` output file as illustrated in the previous section (via `grep`) and calling HyperMRs separately for each context.

Partially methylated regions (PMRs): The `hmr` program also has the option of directly identifying partially methylated regions (PMRs), not to be confused with partially methylated domains (see below). These are contiguous intervals where the methylation level at individual sites is close to 0.5. This should also not be confused with regions that have allele-specific methylation (ASM) or regions with alternating high and low methylation levels at nearby sites. Regions with ASM are almost always among the PMRs, but most PMRs are not regions of ASM. The `hmr` program is run with the same input but a different optional argument to find PMRs:

```
$ hmr -partial -o Human_ESC.pmr Human_ESC.meth
```

Partial methylation in cancer samples (AKA PMDs): Huge genomic blocks with abnormal hypomethylation have been extensively observed in human cancer methylomes and more recently in extraembryonic tissues like the placenta. These domains are characterized by enrichment in intergenic regions or Lamina associated domains (LAD), which are usually hypermethylated in normal tissues. Partially methylated domains (PMDs) are not homogeneously hypomethylated as in the case of HMRs, and contain focal hypermethylation at specific sites. PMDs are large domains with sizes ranging from 10kb to over 1Mb. Hidden Markov Models can also identify these larger domains. The program `pmd` is provided for their identification, and can be run as follows:

```
$ pmd -o Human_ESC.pmd Human_ESC.meth
```

The program calculates in nonoverlapping bins the total methylated and unmethylated read counts. The default bin size is 1000bp, and users can customize the value by specifying the option `-b`. A good way to compare PMDs across samples is to standardize the average number of observations per bin by adjusting for per-sample coverage. The bin-level read counts are modeled with a 2-state HMM in the same form as the model used for HMR detection. The sequence of genomic bins is segmented into hypermethylation and partial-methylation domains, where the latter are the candidate PMDs. Further processing of candidate PMDs includes trimming the two ends of a domain to the first and last CpG positions, and merging candidates that are "close" to each other. Currently, we are using $2 \times bin\_size$ as the merging distance. Development in later versions of the `pmd` program will include randomization procedures for choosing merging distance.

In general, the presence of a single HMR wouldn't cause the program to report a PMD in that region. However, in cases where a number of HMRs are close to each other, such as the promoter HMRs in a gene cluster, the `pmd`

program might report a PMD covering those HMRs. Users should be cautious with using such PMD calls in their further studies. In addition, not all methylomes have PMDs, some initial visualization or summary statistics can be of help in deciding whether to use `pmd` program on the methylome of interest.

In addition, calling HMRs in samples with PMDs can be difficult: PMDs can obscure the sites we are trying to identify by providing an alternative foreground methylation state to the focused, very low methylation typically at promoter regions. A good workaround for this is to call PMDs first, and then call HMRs separately inside and outside of PMDs. This ensures that the foreground methylation state learned by the HMM in both types of background is the focused hypomethylation at CpG islands and promoter regions.

3.2 Differential Methylation

This section explains how to calculate regions of differential methylation between a pair of methylomes or two groups of methylomes. Correspondingly, MethPipe provides two methods for differential methylation (DM) analysis. The first method is designed for computing DM regions between a pair or two small groups of methylomes; it is implemented in the programs `methdiff` and `dmr`. The second method is based on the beta-binomial regression and is appropriate for analysis of datasets containing a larger number of methylomes; it is implemented in the program `radmeth`.

Here we assume that all methylomes correspond to the same genome assembly. To compare methylomes based on different assemblies of the same species or methylomes of different species, they must first be converted to one common genome assembly (see section **??**).

3.2.1 DM analysis of a pair of methylomes or two small groups of methylomes

Comparing a pair of methylomes: Suppose that we want to compare two methylomes: `Human_ESC.meth` and `Human_NHFF.meth`. We start out by calculating the differential methylation score of each CpG site using the `methdiff` program:

```
$ methdiff -o Human_ESC_NHFF.methdiff Human_ESC.meth Human_NHFF.meth
```

Here are the first few lines of the output:

```
$ head -n 4 Human_ESC_NHFF.methdiff
chr1 3000826 + CpG 0.609908 16  7 21 11
chr1 3001006 + CpG 0.874119 21 18 15 22
chr1 3001017 + CpG 0.888384 20 19 15 25
chr1 3001276 + CpG 0.010825  3 20 12 16
```

The first four columns are the same as the methcounts input. The fifth column gives the probability that the methylation level at each given site is lower in `Human_NHFF.meth` than in `Human_ESC.meth`. (For the other direction, you can either swap the order of the two input files or just subtract the probability from 1.0.) The method used to calculate this probability is detailed in Altham (1971) [**?**], and can be thought of as a one-directional version of Fisher's exact test. The remaining columns in the output give the number of methylated reads of each CpG in NHFF, number of unmethylated reads in NHFF, number of methylated reads in ESC, and number of unmethylated reads in ESC, respectively.

With differential methylation scores and HMRs for both methylomes available (see Section **??** for instructions on computing HMRs), DM regions (DMRs) can be calculated with the `dmr` program. This program uses HMR fragments to compute DMRs.

```
$ dmr Human_ESC_NHFF.methdiff Human_ESC.hmr Human_NHFF.hmr \
      DMR_ESC_lt_NHFF DMR_NHFF_lt_ESC
```

The DMRs are output to files `DMR_ESC_lt_NHFF` and `DMR_NHFF_lt_ESC`. The former file contains regions with lower methylation in `Human_ESC.meth` while the latter has regions with lower methylation in `Human_NHFF.meth`. Let's take a look at the first few lines of `DMR_ESC_lt_NHFF`:

```
$ head -n 5 DMR_ESC_lt_NHFF
chr1 3539447 3540231 X:12  0 +
chr1 4384880 4385117 X:6   1 +
chr1 4488269 4488541 X:3   2 +
chr1 4603985 4604344 X:10  2 +
chr1 4760070 4760445 X:8   1 +
```

The first three columns are the genomic coordinates of DMRs. The fourth column contains the numer of CpG sites that the DMR spans, and the fifth column contains the number of significantly differentially methylated CpGs in the DMR. So, the first DMR spans 12 CpGs, but contains no significantly differentially methylated sites, while the second DMR spans 6 CpGs and contains just one significantly differentially methylated CpG site.

We recommend filtering DMRs so that each one contains at least some CpGs that are significantly differentially methylated. This can be easily done with the `awk` utility, available on virtually all Linux and Mac OS systems. For example, the following command puts all DMRs spanning at least 10 CpGs and having at least 5 significantly differentially methylated CpGs into a file `DMR_ESC_lt_NHFF.filtered`.

```
$ awk -F "[:\t]" '$5 >= 10 && $6 >= 5 {print $0}' DMR_ESC_lt_NHFF \
       > DMR_ESC_lt_NHFF.filtered
```

Comparing two small groups of methylomes To compare two small groups of methylomes, one should combine the methylomes in each group and then compute DMRs for the resulting pair of methylomes as described above. The methylomes can be combined using the program `merge-methcounts` (see section **??**).

3.2.2 DM analysis of two larger groups of methylomes

The DM detection method described in this section is based on the beta-binomial regression. We recommend using this method when more than three replicates are available in each group. For rapid differential methylation analysis the regression-based method should be run on a computing cluster with a few hundred available nodes, in which case it takes approximately a few hours to process a dataset consisting of 30-50 WGBS samples. The analysis can be also performed on a personal workstation, but it will take substantially longer. Note that the actual processing time depends on the coverage of each methylome, the number of sites analyzed, and the number of methylomes in the dataset.

Generating proportion tables: The first step in the differential methylation analysis is to assemble a proportion table containing read proportions for all target methylomes. MethPipe includes a program `merge_methcounts` to generate a proportion table from the given collection of methylomes. Suppose that we want to compare methylomes named `control_a.meth`, `control_b.meth`, `control_c.meth` to the methylomes `case_a.meth`, `case_b.meth`, `case_c.meth`. The proportion table can be created with the following command:

```
$ merge-methcounts -t control_a.meth control_b.meth control_c.meth \
                        case_a.meth case_b.meth case_c.meth > proportion_table.txt
```

This is what `proportion_table.txt` looks like:

```
$ head -n 5 proportion_table.txt
control_a control_b control_c case_a case_b case_c
chr1:108:+:CpG  9 6 10 8  1  1  2  2  2 1 14 1
chr1:114:+:CpG 17 7 10 0 14  3  5  1  9 1  7 1
chr1:160:+:CpG 12 8 10 5 17  4 15 14 13 6  4 4
chr1:309:+:CpG  1 1  1 0 17 12 12  8  2 1 19 8
```

As indicated in the header, this proportion table contains information about 6 methylomes: 3 controls and 3 cases. Each row of the table contains information about a CpG site and a proportion of reads mapping over this site in each methylome. For example, the first row describes a cytosine within a CpG site located on chromosome 1 at position 108. This site is present in 9 reads in the methylome `control_a` and is methylated in 6 of them. Note that `merge-methcounts` adds methylomes into the proportion table in the order in which they are listed on the command line.

Design matrix: The next step is to specify the design matrix, which describes the structure of the experiment. For our running example, the design matrix looks like this:

```
$ cat design_matrix.txt
base case
control_a 1 0
control_b 1 0
control_c 1 0
case_a 1 1
case_b 1 1
case_c 1 1
```

The design matrix shows that samples in this dataset are associated with two factors: base and case. The first column corresponds to the base factor and will always be present in the design matrix. Think of it as stating that all samples have the same baseline mean methylation level. To distinguish cases from controls we add another factor case (second column). The 1's in this column correspond to the samples which belong to the cases group. You can use this design matrix as a template to create design matrices for two-group comparisons involving arbitrary many methylomes.

After creating the proportion table and the design matrix, we are now ready to start the methylation analysis. It consists of (1) regression, (2) combining significance, and (3) multiple testing adjustment steps.

Regression: Suppose that the `proportion_table.txt` and `design_matrix.txt` are as described above. The regression step is run with the command

```
$ radmeth regression -factor case design_matrix.txt proportion_table.txt > cpgs.bed
```

The `-factor` parameter specifies the factor with respect to which we want to test for differential methylation. The test factor is `case`, meaning that we are testing for differential methylation between cases and controls. The output file `cpgs.bed` has the same format `methdiff`. The p-value (5-th column) is given by the log-likelihood ratio test. A p-value of -1 means that the test was not performed: either due to zero coverage over all case or control samples, or because the methylation level is identical in all samples: both cases in which the regression would fail. We do not recommend using p-values generated by `radmeth regression` directly, instead we adjust the p-value of each CpG site based on the p-values of the neighboring CpGs.

Combining significance and adjusting for multiple testing: Both of these steps are performed simultaneously. Given the cpgs.bed file from the previous step, run

```
$ radmeth adjust -bins 1:200:1 cpgs.bed > cpgs.adjusted.bed
```

Here, the only required parameter, besides the input file, is `-bins` whose value is set to 1:200:1 (which is also the default value). This means that for each $n = 1, 2, ...199$, `radmeth adjust` computes the correlation between p-values of CpGs located at distance n from each other. These correlations are used during significance combination step. In addition, bin sizes determine the window for combining significance. In contrast, if `-bins` is set to $1 : 15 : 5$, then the correlation is computed separately for p-values corresponding to CpGs at distances $[1, 5)$, $[5, 10)$, and $[10, 15)$ from one another.

The first five columns and the last four columns of `radmeth adjust` have the same meaning as those output by `radmeth regression`. The 6-th column gives the modified p-value based on the original p-value of the site and the p-values of its neighbors. The 7-th column gives the FDR-corrected p-value. Then the last four columns corresponds to the total read counts and methylated read counts of the case group and control group, respectively.

Here is what the 'cpgs.adjusted.bed' le looks like for our example dataset:

```
$ head -n 5 cpgs.adjusted.bed
chr1 108 + CpG 0.157971 0.099290 0.353466 18  4 20 15
chr1 114 + CpG 0.559191 0.099290 0.353466 21  3 41 10
```

```
chr1 160 + CpG 0.095112 0.099290 0.353466 32 24 39 17
chr1 309 + CpG 0.239772 0.122248 0.368902 33 17 19 13
chr1 499 + CpG 0.770140 0.204467 0.419872 43 22 29 15
```

Individual differentially methylated sites: After completing the previous steps, individual differentially methylated sites can be obtained with 'awk'. To get all CpGs with FDR-corrected p-value below 0.01, run

```
$ awk '$7 <= 0.01 "{ print $0; $}"' cpgs.adjusted.bed > dm_cpgs.bed
```

Differentially methylated regions: It is possible to further join individually differentially methylated CpGs into differentially methylated regions. This can be achieved with the command

```
$ radmeth merge -p 0.01 cpgs.adjusted.bed > dmrs.bed
```

The current algorithm is conservative: it joins neighboring differentially methylated sites with p-value below 0.01 (set by the `-p` parameter). The output format is

```
chrom start end dmr num-sites meth-diff
```

where `num-sites` and `meth-diff` are the number of significantly differentially methylated CpGs in the DMR and the estimated methylation difference. For our example, the output looks like this:

```
$ head -n 5 dmrs.bed
chr1  57315  57721 dmr 10 -0.498148
chr1  58263  59009 dmr 27 -0.521182
chr1 138522 139012 dmr 13 -0.443182
chr1 149284 149444 dmr  7 -0.430453
chr1 274339 275254 dmr 18 -0.520114
```

3.3 Allele-specific methylation

Allele-specific methylation (ASM) occurs when the same cytosine is differentially methylated on the two alleles of a diploid organism. ASM is a major mechanism of genomic imprinting, and aberrations can lead to disease. Included in `methpipe` are three tools to analyze ASM: `allelicmeth`, `amrfinder`, and `amrtester`. All of these programs calculate the probability of ASM in a site or region by counting methylation on reads and analyzing the dependency between adjacent CpGs, and therefore it is recommended that any samples analyzed have at least 10$\times$ coverage and 100bp reads for the human genome.

3.3.1 Epiread Format

All programs that calculate statistics related to ASM must take read distribution into account. Because mapped read (.mr) files are unwieldy and in some cases very large, we defined an intermediate format, epiread, to encapsulate read information in a more efficient manner. Epiread format consists of three columns. The first column is the chromosome of the read, the second is the numbering order of the first CpG in the read, and the last is the CpG-only sequence of the read, leading to a large decrease in size and complexity. The program `methstates` has been provided to convert mapped read files, and an example is shown below:

```
$ methstates -c hg19 -o Human_ESC.epiread Human_ESC.mr
```

3.3.2 Single-site ASM scoring

The program `allelicmeth` calculates allele specific methylation scores for each CpG site. Input files should be the epiread files (`.epiread` suffix) produced in the previous section. In the output file, each row represents a CpG pair made by any CpG and its previous CpG, the first three columns indicate the positions of the CpG site, the fourth column is the name including the number of reads covering the CpG pair, the fifth column is the score for ASM, and the last four columns record the number of reads of four different methylation combinations of the CpG pair: methylated methylated (mm), methylated unmethylated (mu), unmethylated methylated (um), or unmethylated unmethylated (uu). The following command will calculate allele specific methylation scores using the `allelicmeth` component of Methpipe:

```
$ allelicmeth -c hg19 -o Human_ESC.allelic Human_ESC.epiread
```

3.3.3 Allelically methylated regions (AMRs)

The method described here was introduced in [**?**]. The program `amrfinder` scans the genome using a sliding window to identify AMRs. For a genomic interval, two statistical models are fitted to the reads mapped, respectively. One model (single-allele model) assumes the two alleles have the same methylation state, and the other (two-allele model) represents different methylation states for the two alleles. Comparing the likelihood of the two models, the interrogated genomic interval may be classified as an AMR.

The following command shows an example to run the program `amrfinder`.

```
$ amrfinder -o Human_ESC.amr -c hg18 Human_ESC.epiread
```

There are several options for running `amrfinder`. The `-b` switches from using a likelihood ratio test to BIC as the criterion for calling an AMR. The `-i` option changes the number of iterations used in the EM procedure when fitting the models. The `-w` option changes the size of the sliding window, which is in terms of CpGs. The default of 10 CpGs per window has worked well for us. The `-m` indicates the minimum coverage per CpG site required for a window to be tested as an AMR. The default requires 4 reads on average, and any lower will probably lead to unreliable results. AMRs are often fragmented, as coverage fluctuates, and spacing between CpGs means their linkage cannot be captured by the model. The `-g` parameter is used to indicate the maximum distance between any two identified AMRS; if two are any closer than this value, they are merged. The default is 1000, and it seems to work well in practice, not joining things that appear as though they should be distinct. In the current version of the program, at the end of the procedure, any AMRs whose size in terms of base-pairs is less than half the "gap" size are eliminated. This is a hack that has produced excellent results, but will eventually be eliminated (hopefully soon).

Finally, the `-C` parameter specifies the critical value for keeping windows as AMRs, and is only useful when the likelihood ratio test is the used; for BIC windows are retained if the BIC for the two-allele model is less than that for the single-allele model. `amrfinder` calculates a false discovery rate to correct for multiple testing, and therefore most p-values that pass the test will be significantly below the critical value. The `-h` option produces FDR-adjusted p-values according to a step-up procedure and then compares them directly to the given critical value, which allows further use of the p-values without multiple testing correction. The `-f` omits multiple testing correction entirely by not applying a correction to the p-values or using a false discovery rate cutoff to select AMRs.

In addition to `amrfinder`, which uses a sliding window, there is also the `amrtester` program, which tests for allele-specific methylation in a given set of genomic intervals. The program can be run like this:

```
$ amrtester -o Human_ESC.amr -c hg19 intervals.bed Human_ESC.epiread
```

This program works very similarly to `amrfinder`, but does not have options related to the sliding window. This program outputs a score for each input interval, and when the likelihood ratio test is used, the score is the p-value, which can easily be filtered later.

3.4 Consistent estimation of hydroxymethylation and methylation levels

If you are interested in estimating hydroxymethylation level and have any two of Tet-Assisted Bisulfite sequencing (TAB-seq), oxidative bisulfite sequencing (oxBS-seq) and BS-seq data available, you can use `mlml` [**?**] to perform consistent and simultaneous estimation.

The input file format could be the default `methcounts` output format described in Section **??**, or BED format file with 6 columns as the example below:

```
chr1    3001345 3001346 CpG:9   0.777777777778  +
```

Here the fourth column indicates that this site is a CpG site, and the number of reads covering this site is 9. The fifth column is the methylation level of the CpG site, ranging from 0 to 1. Note that all input files must be sorted. Assume you have three input files ready: `meth_BS-seq.meth`, `meth_oxBS-seq.meth` and `meth_Tab-seq.meth`. The following command will take all the inputs:

```
$ mlml -v -u meth_BS-seq.meth -m meth_oxBS-seq.meth \
         -h meth_Tab-seq.meth -o result.txt
```

If only two types of input are available, e.g. `meth_BS-seq.meth` and `meth_oxBS-seq.meth`, then use the following command:

```
$ mlml -u meth_BS-seq.meth -m meth_oxBS-seq.meth \
         -o result.txt
```

In some cases, you might want to specify the convergence tolerance for EM algorithm. This can be done through `-t` option. For example:

```
$ mlml -u meth_BS-seq.meth -m meth_oxBS-seq.meth \
         -o result.txt -t 1e-2
```

This command will make the iteration process stop when the difference of estimation between two iterations is less than 10^{-2}. The value format can be scientific notation, e.g. 1e-5, or float number, e.g. 0.00001.

The output of `mlml` is tab-delimited format. Here is an example:

```
chr11 15 16 0.166667 0.19697 0.636364 0
chr12 11 12 0.222222 0 0.777778 2
```

The columns are chromosome name, start posistion, end position, 5-mC level, 5-hmC level, unmethylated level and number of conflicts. To calculate the last column, a binomial test is performed for each input methylation level (can be 2 or 3 in total depending on parameters). If the estimated methylation level falls out of the confidence interval calculated from input coverage and methylation level, then such event is counted as one conflict. It is recommended to filter estimation results based on the number of conflicts; if more conflicts happens on one site then it is possible that information from such site is not reliable.

3.5 Computing average methylation level in a genomic interval

One of the most common analysis tasks is to compute the average methylation level through a genomic region. The `roimethstat` program accomplishes this. It takes a sorted `methcounts` output file and a sorted BED format file of genomic "regions of interest" (hence the "roi" in `roimethstat`). If either file is not sorted by (chrom,end,start,strand) it can be sorted using the following command:

```
$ LC_ALL=C sort -k 1,1 -k 3,3n -k 2,2n -k 6,6 \
         -o regions_ESC.meth.sorted regions_ESC.meth
```

From there, `roimethstat` can be run as follows:

```
$ roimethstat -o regions_ESC.meth regions.bed Human_ESC.meth.sorted
```

The output format is also 6-column BED, and the score column now takes the average methylation level through the interval, weighted according to the number of reads informing about each CpG or C in the methylation file. The 4th, or "name" column encodes several other pieces of information that can be used to filter the regions. The original name of the region in the input regions file is retained, but separated by a colon (`:`) are, in the following order, (1) the number of CpGs in the region, (2) the number of CpGs covered at least once, (3) the number of observations in reads indicating in the region that indicate methylation, and (4) the total number of observations from reads in the region. The methylation level is then (3) divided by (4). Example output might look like:

```
chr1  3011124  3015902  REGION_A:18:18:105:166  0.63253  +
chr1  3015904  3016852  REGION_B:5:5:14:31  0.451613  +
chr1  3017204  3017572  REGION_C:2:2:2:9  0.222222  -
chr1  3021791  3025633  REGION_D:10:10:48:73  0.657534  -
chr1  3026050  3027589  REGION_E:2:4:4:32:37  0.864865  -
```

Clearly if there are no reads mapping in a region, then the methylation level will be undefined. By default `roimethstat` does not output such regions, but sometimes they are helpful, and using the `-P` flag will force `roimethstat` to print these lines in the output (in which case every line in the input regions will have a corresponding line in the output).

It is routinely useful to compute the average methylation state across a large number of target regions (for instance, refseq genes or all LINE retrotransposons). `roimethstat` performs a binary search on the methcounts file to find the CpGs associated with each target region and loads only those CpGs to save memory. However, with many target regions this can be tedious. The -L option loads all lines of the methcounts file into memory, which saves time at the expense of an increased memory requirement.

3.6 Computing methylation entropy

The concept of Entropy was introduced into epigenetics study to characterize the randomness of methylation patterns over several consecutive CpG sites [**?**]. The`methentropy` program processes epireads and calculates the methylation entropy value in sliding windows of specified number of CpGs. Two input files are required, including the directory containing the chromosome fasta files, and an epiread file as produced by `methstates` program. The input epiread file needs to be sorted, first by chromosome, then by position.

```
$ LC_ALL=C sort -k1,1 -k2,2g Human_ESC.epiread -o Human_ESC.epiread.sorted
```

Use the `-w` option to specify the desired number of CpGs in the sliding window; if unspecified, the default value is 4. In cases where symmetric patterns are considered the same, specify option `-F`, this will cause the majority state in each epiread to be forced into "methylated", and the minority to "unmethylated". The processed epireads will then be used for entropy calculation. To run the program, type command:

```
$ methentropy -w 5 -v -o Human_ESC.entropy hg18 Human_ESC.epiread.sorted
```

The output format is the same as `methcount` output. The first 3 columns indicate the genomic location of the center CpG in each sliding window, the 5th column contains the entropy values, and the 6th column shows the number of reads used for each sliding window. Below is an output example.

```
chr1 483 + CpG 2.33914 27
chr1 488 + CpG 2.05298 23
chr1 492 + CpG 1.4622 24
chr1 496 + CpG 1.8784 35
```

3.7 Notes on data quality

The performance of our tools to identify higher-level methylation features (HMR, HyperMR, PMDs and AMR) depends on the underlying data quality. One major factor is coverage. Based on our experience, HMR detection using our method is acceptable above 5x coverage, and we recommend 10x for reliable results. Our method for identifying HyperMRs is similar to HMR-finding method, and the above statement holds. The required coverage therefore is even lower since the PMD-finding method internally works by accumulating CpGs in fixed-length bins. We feel $\sim$3x is sufficient. The AMR method depends on both coverage and read length. Datasets with read length around 100bp and mean coverage above 10x are recommended for the AMR method. Another important measure of data quality is the bisulfite conversion rate. Since most datasets have pretty good bisulfite conversion rate (above 0.95), our tools does not explicitly correct fo the conversion rate.

4 Methylome visualization

4.1 Creating UCSC Genome Browser tracks

To view the methylation level or read coverage at individual CpG sites in a genome browser, one needs to create a `bigWig` format file from a `*_.meth` file, which is the output of the `methcounts` program. A methcounts file would look like this:

```
chr1 468 469 CpG:30 0.7 +
chr1 470 471 CpG:29 0.931034 +
chr1 483 484 CpG:36 0.916667 +
chr1 488 489 CpG:36 1 +
```

The first 3 columns shows the physical location of each CpG sites in the reference genome. The number in the 4th column indicates the coverage at each CpG site. The methylation level at individual CpG sites can be found in the 5th column. To create methylation level tracks or read coverage tracks, one can follow these steps:

1. Download the `wigToBigWig` program from UCSC genome browser's directory of binary utilities (`http://hgdownload.cse.ucsc.edu/admin/exe/`).
2. Use the `fetchChromSizes` script from the same directory to create the `*.chrom.sizes` file for the UCSC database you are working with (e.g. hg19). Note that this is the file that is referred to as `hg19.chrom.sizes` in step 3.
3. To create a `bw` track for methylation level at single CpG sites, convert the methcounts file to bed format using:

```
$ awk -v OFS="\t" '{print $1, $2, $2+1, $4":"$6, $5, $3}' \
      Human_ESC.meth > Human_ESC.meth.bed
```

4. To create a `bw` track from the bed format methcounts output, modify and use the following command:

```
$ cut -f 1-3,5 Human_ESC.meth.bed | \
     wigToBigWig /dev/stdin hg19.chrom.sizes Human_ESC.meth.bw
```

To create a `bw` track for coverage at single CpG sites, modify and use the following command:

```
$ tr ':' '[Ctrl+v Tab]' < Human_ESC.meth.bed | cut -f 1-3,5 | \
     wigToBigWig /dev/stdin hg19.chrom.sizes Human_ESC.reads.bw
```

Note that if the `wigToBigWig` or `fetchChromSizes` programs are not executable when downloaded, do the following:

```
$ chmod +x wigToBigWig
$ chmod +x fetchChromSizes
```

You might also want to create `bigBed` browser tracks for HMRs, AMRs, PMDs, or DMRs. To do so, follow these steps:

1. Download the `bedToBigBed` program from the UCSC Genome Browser directory of binary utilities (`http://hgdownload.cse.ucsc.edu/admin/exe/`).
2. Use the `fetchChromSizes` script from the same directory to create the `*.chrom.sizes` file for the UCSC database you are working with (e.g. hg19). Note that this is the file that is referred to as `hg19.chrom.sizes` in step 3.
3. Modify and use the following commands: PMDs, HMRs and AMRs may have a score greater than 1000 in the 5th column, in which case bedToBigBed will output an error. Also, `*_HMR.bed` files may have non-integer score in their 5th column. The following script rounds the 5th column and prints 1000 if the score is bigger than 1000:

```
$ awk -v OFS="\t" '{if($5>1000) print $1,$2,$3,$4,"1000"; \
          else print $1,$2,$3,$4,int($5) }' Human_ESC.hmr \
          > Human_ESC.hmr.tobigbed

 bedToBigBed Human_ESC.hmr.tobigbed hg19.chrom.sizes Human_ESC.hmr.bb
```

   In the above command, since the HMRs are not stranded, we do not print the 6th column. Keeping the 6th column would make all the HMRs appear as though they have a direction – but it would all be the + strand. To maintain the 6th column, just slightly modify the above `awk` command:

```
$ awk -v OFS="\t" '{if($5>1000) print $1,$2,$3,$4,"1000",$6; \
          else print $1,$2,$3,$4,int($5),$6 }' Human_ESC.hmr \
          > Human_ESC.hmr.tobigbed
```

4. Generate the .bb track using the command below:

```
$ bedToBigBed Human_ESC.hmr.tobigbed hg19.chrom.sizes Human_ESC.hmr.bb
```

4.2 Converting browser tracks to `methcounts` format

All tracks in MethBase are available to download through sample description page (See `http://smithlabresearch.org/software/methbase/` for details). We provide a python script for converting browser tracks back to original `methcounts` file for downstream analyses of users' requirement. For each methylome on MethBase, there are two tracks that are necessary for the conversion: the track for methylation levels (`.meth.bw`) and the track for sequencing coverage (`.read.bw`). One needs to download these tracks following the links in the description page of each methylome. An external program is required for converting `bigWig` files to BEDGraph format, whose name is `bigWigToBedGraph` and can be found in `http://hgdownload.cse.ucsc.edu/admin/exe/` under corresponding OS environment. After these required files are ready, user may follow the following steps to create a `methcounts` file.

1. Find the script located in `METHPIPE_ROOT/src/utils` with the name `bigWig_to_methcounts.py`. Here we use `METHPIPE_ROOT` to represent the path to the installation location of `methpipe`.
2. Locate `bigWigToBedGraph`. Type the following command:

```
$ which bigWigToBedGraph
```

If it returns nothing, then you need to find a absolute path to the program such as

```
/home/user/programs/bigWigToBedGraph
```

3. Run the script like below:

```
$ python ./bigWig_to_methcounts.py -m Human_ESC.meth.bw \
          -r Human_ESC.read.bw -o Human_ESC.meth -p PATH_TO_PROGRAM
```

Use the path found in Step 2 as `PATH_TO_PROGRAM` for parameter `-p`.

5 Auxiliary tools

5.1 Count number of lines in a big file

When working with next-generation sequencing data, researchers often handle very large files, such as FASTQ files containing raw reads and `*.mr` files containing mapped reads. `lc_approx` is an auxiliary tool designed to approximate the number of lines in a very large file by counting the number of lines in a small, randomly chosen chunk from the big file and scaling the estimate by file size. For example, in order to estimate the number of reads in a FASTQ file `s_1_1_sequence.fq`, run

```
$ lc_approx  s_1_1_sequence.fq
```

It will return the approximate number of lines in this file and by dividing the above number by 4, you get the approximate number of reads in that file. The `lc_approx` can be hundreds of times faster than the unix tool `wc -l`.

5.2 Mapping methylomes between species

Mapping methylomes between species builds on the `liftOver` tool provided by UCSC Genome Browser `http://genome.ucsc.edu/cgi-bin/hgLiftOver`. However it is time consuming to convert each `methcounts` output file from one assembly to another using the UCSC `liftOver` tool, given that they all should have the same locations but different read counts. Therefore, we use `liftOver` to generate an index file between two assemblies, and provide the `fast-liftover` tool.

Suppose we have downloaded the `liftOver` tool and the chain file `mm9ToHg19.over.chain.gz` from the UCSC Genome Browser website. If we have a `methcounts` file `Mouse_BCell_mm9.meth` of CpG sites or all cytosines in mm9. Entries in `Mouse_BCell_mm9.meth` look like

```
chr1  3005765  +  CpG  0.166667  6
chr1  3005846  +  CpG  0.5        10
chr1  3005927  +  CpG  0         9
```

We would like to lift it over to the human genome hg19, and generate an index file `mm9-hg19.index` to facilitate later lift-over operations from mm9 to hg19, and keep a record of unlifted mm9 cytosine positions in the file `mm9-hg19.unlifted`. First, prepare the input `mm9_cpg.bed` file for `liftOver` into the following BED format:

```
chr1  3005765  3005766  chr1:3005765:3005766:+  0  +
chr1  3005846  3005847  chr1:3005846:3005847:+  0  +
chr1  3005927  3005928  chr1:3005927:3005928:+  0  +
```

Note that the 4th column is the genomic location data linked with colons.

Then, run UCSC Genome Browser tool `liftOver` as follows:

```
$ liftOver mm9_cpg.bed mm9ToHg19.over.chain.gz  mm9-hg19.index mm9-hg19.unlifted
```

The generated index file `mm9-hg19.index` will be a BED format file in hg19 coordinates, with entries like

```
chr8 56539820 56539821  chr1:3005765:3005766:+ 0 -
chr8 56539547 56539548  chr1:3005846:3005847:+ 0 -
chr8 56539209 56539210  chr1:3005927:3005928:+ 0 -
```

where the 4th column contains the genomic position of the cytosine site in mm9 coordinates.

After the index file is generated, we can use the `fast-liftover` program on any mm9 `methcount` file to lift it to hg19:

```
$ fast-liftover -i mm9-hg19.index -f SAMPLE_mm9.meth -t SAMPLE_hg19.meth.lift -v
```

The `-p` option should be specified to report positions on the positive strand of the target assembly.

Before using the lifted methcount file, make sure it is sorted properly.

```
$ LC_ALL=C sort -k1,1 -k2,2g -k3,3 SAMPLE_hg19.meth.lift -o SAMPLE_hg19.meth.sorted
```

The `liftOver` program may report multiple mm9 sites mapped to a same position in hg19.

In this situation, we may either collapse read counts at those mm9 sites, or keep the data for only one mm9 site. We can use the `lift-filter` program to achieve these two options. Use

```
$ lift-filter -o SAMPLE_hg19.meth  SAMPLE_hg19.meth.sorted -v
```

to merge data from mm9 sites lifted to the same hg19 position. Use the option `-u` to keep the first record of duplicated sites.