The Smithlab DNA Methylation Data Analysis Pipeline (MethPipe)

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

September 6, 2013

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

Contents

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 BS-seq 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 adenosine (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 `rmapbs` 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 rmapbs is to assume that reads are T-rich and map accordingly. To change the mapping to suit A-rich reads, add the `-A` option.

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 following two `methpipe` components, `bsrate` and `methcounts`.

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

Adaptor sequences must be supplied to `rmapbs` through the `-C` option. Keep in mind that if the adaptor 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 `rmapbs`.

Single-end reads: When working with data from a single-end sequencing experiment, you will have T-rich reads only. `rmapbs` expects T-rich reads as a default and so you do not have use the `-A` option to change mapping parameters. Execute the following command to map all of your single-end reads with `rmapbs`:

```
$ ./rmapbs -c hg18 -o Human_NHFF.mr Human_NHFF.fastq
```

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. The program `rmapbs-pe` program is used to map T-rich reads and A-rich reads simultaneously. Run the following command to map two reads files from a paired-end sequencing experiment:

```
$ ./rmapbs-pe -c hg18 -o Human_ESC.mr Human_ESC_1.fastq Human_ESC_2.fastq
```

In brief, what happends internally in `rmapbs-pe` is as follows. `rmapbs-pe` 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 `rmapbs-pe` 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 adaptors. 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, `rmapbs-pe` will throw each mate individually if its mapping position is unambiguous.

Mapping reads in a large file: Mapping reads often takes a while, and mapping reads from BS-seq 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. I will typically map 3M reads at a time, and this takes at most 1.5GB of memory for the human genome and with 100nt reads. If all computing nodes can read the large input file (for example, through NFS), you may use the option `-T s` and `-N n` to instruct `rmapbs` or `rmapbs-pe` to map `n` reads starting from the s^{th} reads in the input file. For example, with following command

```
$ ./rmapbs -c hg18 -o Human_NHFF.mr Human_NHFF.fastq -T 1 -N 1000000
```

rmapbs will map the first million reads in the input file.

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. Not a problem, just use the cat command.

Alternative mappers: We described how to map bisulfite treated reads with the rmapbs program. Users may also use alternative mappers, for example, BSSeeker, which uses three-alphabeta stragy, and BSMAP, which allows 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 .sam format, run

```
$ ./to-mr -o Human_NHFF.mr -m bsmap Human_NHFF.sam
```

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.

The program to-mr only supports .sam format by default installation, and .bam format support is enabled with BamTools properly installed. BamTools can be found at: https://github.com/pezmaster31/bamtools. User must specify the path to BamTools in compile. For example, if BamTools is installed at /home/user/bamtools, then the path should be added in the compile command as

```
$ make all BAMTOOLS_ROOT=/home/user/bamtools
```

After successful compilation, user can add -b to convert .bam files. For pair-end data, to-mr will automatically clip the two ends to one fragment.

2.2 Merging libraries and removing duplicates

Before calculating methylation level, you should now remove read duplicates, or reads that were mapped to the same genomic location. 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 chromosom, 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. Please refer to 5 for recommended practices to organize a project with multiple runs and/or libraries.

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, bisulphite 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 mitochondrial or chloroplast genomes which are believed not to be methylated, (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 C to (C + T) at these positions. If the bisulfite reaction were 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 hg18 -o Human_ESC.bsrate Human_ESC.mr
$ ./bsrate -c hg18 -o Human_NHFF.bsrate Human_NHFF.mr
```

The `bsrate` program requires that the input be sorted so that reads mapping to the same chromosome are contiguous. 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.

When using `bsrate` on paired-end reads that have not yet gone through the `clipmates` stage of the pipeline, the second-end reads must be treated separately as they will still be A-rich:

```
$ ./bsrate -c hg18 -o s_1_1_sequence.bsrate s_1_1_sequence.mr
$ ./bsrate -c hg18 -o s_1_2_sequence.bsrate -A s_1_2_sequence.mr
```

If you are using reads from an unmethylated spike-in or reads mapping to mitochondria, then there is an option to use all Cs, including those at CpG sites:

```
$ 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.

2.4 Computing single-site methylation levels

The `methcounts` program takes the mapped reads and produces the methylation level for each genomic CpG or for all Cs if specified. The input is in MappedRead format, and the reads should be sorted according to (chrom, end, start, strand). If your reads are not sorted, run:

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

Since `methcounts` can only take one input file, if you have multiple you can merge them using the `-m` option to the `sort` program:

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

Counting only CpG sites: The methylation level for every CpG site at single base resolution is estimated as a probability based on the ratio of methylated to total reads mapped to that loci. Since CpG methylation is symmetric, reads mapped to both strands are used to produce a single estimate for the CpG site. To compute methylation levels at each CpG site you can use commands as following:

```
$ ./methcounts -c hg18 -o Human_ESC.meth \
               -S Human_ESC.methstats Human_ESC.mr
$ ./methcounts -c hg18 -o Human_NHFF.meth \
               -S Human_NHFF.methstats Human_NHFF.mr
```

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).

The output file contains one line per CpG site. The first column is the chromosome. The second is the location of the CpG site in the positive strand. The 3rd column indicates the strand, always being + when only CpG sites are considered. The 4th column is the sequence context of that site. 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.

Counting all cytosines: While methylation usually exists in the CpG contexts, in certain mammalian cells and most plant cells, cytosines in other sequence contexts, such as CHG or CHH (where H denotes adenines, thymines or cytosines), may also be methylated. This type of methylation is asymmetric since the cytosines on the complementary strand do not necessarily have the same methylation status. The methylation level for each cytosine loci is estimated individually with only reads mapped to the strand where that cytosine is located. The estimate of the methylation level is given by the number of methylated reads mapped to that cytosine divided by the total number of reads. Since most mammalian methylation occurs in the context of CpG dinucleotides, `methcounts` calculates methylation levels for only CpG sites by default; to calculate methylation levels for all cytosines in an asymmetric way, add the `-N` option like the following,

```
$ ./methcounts -N -c hg18 -o Human_ESC_All.meth \
               -S Human_ESC_Meth_All.methstats Human_ESC.mr
```

The output file contains one line per cytosine site (see below for an excerpt from one `methcounts` output file in an Arabidopsis project). The first two columns give the chromosome and position of that cytosine. The 3rd columns gives the strand. The 4th column indicates the sequence context, which can be either CG, CHG or CHH. The first C corresponds to the cytosine of interest and the remaining letters are bases following that cytosine in the same strand from 5' to 3'. The 5th and 6th columns gives the estimated methylation level and the total number of reads respectively.

```
chr1    839537  -       CG      0.8     5
chr1    839541  -       CHH     0.142857        7
chr1    839544  +       CHH     0       10
chr1    839548  -       CHH     0       7
chr1    839550  -       CHH     0       8
chr1    839551  +       CG      0.75    8
chr1    839552  -       CG      0.75    8
chr1    839554  +       CHH     0       8
chr1    839557  -       CHH     0       7
chr1    839566  +       CG      0.857143        7
chr1    839567  -       CG      0.714286        7
```

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`.

Merging methcounts file from multiple replicates: When working with a BS-seq project with multiple replicates, you may first produce a methcount 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 have higher coverage. Suppose you have the three methcounts files from three different biological replicates, `ASDF_R1/R1.meth`, `ASDF_R2/R2.meth` and `ASDF_R3/R3.meth`. To merge those individual methcounts files, execute

```
$ merge-methcounts ASDF_R1/R1.meth  ASDF_R2/R2.meth \
  ASDF_R3/R3.meth -o ASDF.meth
```

Computation of average methylation level The `levels` program computes average methylation in a methcounts file 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. To run the `levels` program, execute

```
$ levels -o ASDF.levels ASDF.meth
```

3 Methylome analysis

The following tools will analyze much of the information about CpG's 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), and regions with allele-specific methylation (AMRs).

3.1 Hypomethylated and hypermethylated regions (HMRs)

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 above). The following command will work well for identifying mammalian HMRs if there is sufficient coverage in the underlying methylomes:

```
$ ./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. 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.

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 `hmr-plant` 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

```
$ ./hmr-plant -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 +.

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
```

Giant HMRs observed in cancer samples (AKA PMDs): Huge genomic blocks with abnormal hypomethylation have been extensively observed in human cancer methylomes, with high enrichment in intergenic regions or Lamina associated domains (LAD), which are usually hypermethylated in normal tissues. These huge blocks are not homogeneously hypo-methylated in most cases. Focal hypermethylation are also observed within these blocks. Hidden Markov Model can catch these big blocks, and is also sensitive to methylation changes at a smaller scale. As a result, the cancer-specific hypomethylated blocks can be identified with clusters of HMRs given by the `hmr` program. These HMRs are usually longer and closer to each other than normal HMRs. Definition of blocks can be achieved by merging these clustered cancer-specific HMRs.

3.2 Differential methylation between two methylomes

If you are working with more than one methylome, it may be of interest to you to identify regions between your methylomes that have significantly different levels of methylation. To do this, use the programs `methdiff` and `dmr`. Run `methdiff` first since its output serves as the input for `dmr`. Since methylation differences are assessed on a per CpG basis, the methylomes being compared must come from the same genomes. Otherwise, comparisons will not be between orthologous CpG's. If you would like to compare methylomes from different genomes (i.e. human and chimp methylomes), you must first convert the CpG coordinates for one species into their orthologous coordinates for the other species. Additionally, `methdiff` and `dmr` can only compare two methylomes at a time. Each of these programs is explained in more detail in the subsections below.

3.2.1 Differential methylation scores

The program `methdiff` produces a differential methylation score for each CpG in a methylome. This score indicates the probability that the CpG is significantly less methylated in one methylome than the other. The inputs for `methdiff` are the output of `methcounts` for each of the two methylomes being analyzed. The following command calculates differential methylation scores across two methylomes using the `methdiff` component of Methpipe.

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

So in the output file `Human_ESC_NHFF.methdiff` the 5th column indicates 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 those two input files, or just subtract the probability from 1.0. The method used is due to Altham (1971) [**?**], and is like a one-directional version of Fisher's exact test.

3.2.2 Differentially methylated regions (DMRs)

Once differential methylation scores have been calculated, the program `dmr` can be used to identify differentially methylated regions, or DMRs. DMRs are regions where differential methylation scores indicate there are many CpGs with a high probability of being differentially methylated between the two methylomes. `dmr` uses HMR data from the two methylomes and identifies a DMR wherever an HMR exists in one methylome but not the other. It writes the DMRs into two files: one with the HMR in one methylome and another with the HMR in the other. It also writes the total number of CpGs in the DMR and the number of significantly different CpG sites. The following command finds DMRs using the `dmr` component of `methpipe`:

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

3.3 Allele-specific methylation

If you are interested in different methylation states of the two alleles, use the programs `allelicmeth` and `amrfinder`. For each CpG site, `allelicmeth` calculates the probability that the site has allele-specific methylation (ASM). The higher the score, the more likely the CpG site has ASM. `amrfinder` is used to identify allelically methylated regions (AMRs). Both programs make use of the dependency between adjacent CpGs. Therefore, longer reads and higher coverage may improve the accuracy. Also the programs work better in CpG dense regions (*e.g.* CpG islands). Typically, 10$\times$ coverage and 100bp reads can ensure good performance in the human genome.

3.3.1 Allele-specific methylation scores

The program `allelicmeth` calculates allelic-specific methylation scores for each CpG site. Input files should be the mapped reads files (`.mr` suffix) produced previously in the mapping step. 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 hg18 -o Human_ESC.allelicmeth Human_ESC.mr
```

3.3.2 Allelically methylated regions (AMRs)

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 is determined whether or not an AMR. The input files are still mapped reads files, and the output bed files contain all possible AMRs in BED format. The following command shows an example to run the program `amrfinder`.

```
$ ./amrfinder -o Human_ESC.amr -i 10 -w 10 -m 1 -c hg18 -b Human_ESC.mr
$ ./amrfinder -o Human_ESC.amr -c hg18 -b -E Human_ESC.epiread
```

Option `-i` is the maximum iterations allowed in the EM procedure when calculating the likelihood for the two-allele model, and the default value is 10. Option `-w` defines the size of the sliding window using the number of CpGs, and the default value is 10. Option `-m` is the requirement of the minimum reads covering each CpG, and the default value is 1. Option `-b` means to use BIC criterion to compare the two likelihood models, otherwise likelihood ratio test is used for model comparison. There is an option `-E` indicating if the input file is in a special format called "Epiread" format, which consists of three columns. The first column is the chromosome of the read, the second column is the numbering order of the first CpG in the read, and the last column is the CpG-only sequence of the read. Such 'Epiread' format reduces the memory requirement. You can use the program `methstates` to convert the mapped reads file to the epiread format as below:

```
$ ./methstates -c hg18 Human_ESC.mr -o Human_ESC.epiread
```

3.4 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).

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, 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 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: For `*_HMR.bed` files with non-integer score in their 5th column, one needs to round the score to integer value, for example:

   ```
   $ awk -v OFS="\t" '{print $1,$2,$3,$4,int($5)}' Human_ESC.hmr | \
            bedToBigBed /dev/stdin 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.

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

5 Organizing projects with multiple replicates and/or libraries

Often to build a methylome for a particular population of cells one obtains multiple biological replicates. For each of these biological replicates, one may construct more than one bisulfite sequencing library, and these correspond to technical replicates. Finally, a given library might be sequenced multiple times, and a given sequencing run might produce one or more files of reads (for example, corresponding to different tiles or lanes on an Illumina sequencer). We organize data to reflect these aspects of the experiment.

Biological replicates Distinct biological replicates refer to distinct cells from which DNA was extracted. Typically these are obtained form different individuals or different cell cultures. If one is building the "refernce" methylome for a given cell type, then eventually these different biological replicates might be merged. However, the purpose of doing the replicates separately is so that the biological variation can be understood. Therefore, certain tests would usually be done prior to combining the different biological replicates. Fortunately, the methylomes corresponding to different biological replicates can be done at much lower coverage, because testing for biological variation does not require conducting all the kinds of analyses we would want to conduct on the reference methylome. Distinct biological replicates are necessarily distinct technical replicates, and so to combine them we would follow the same procedure as combining technical replicates (see below). If the name of our methylome is `ASDF` then we would organize our biological replicates in a directory structure like this:

```
./ASDF/ASDF_R1/
./ASDF/ASDF_R2/
./ASDF/ASDF_R3/
```

The properties that we would typically associate with a biological replicate include the average methylation levels through different parts of the genome. When comparing biological replicates we might observe some differences in average methylation levels through promoters, for example. We would hope and expect these to be minimal, resulting either from noise due to sampling (when coverage is low) or variation associated with genotype.

Distinct libraries The distinct libraries constitute technical replicates. Each technical replicate may have a different distribution of fragment lengths (as measured using paired-end information or amount of adaptors present in reads), and possibly a different bisulfite conversion rate. Despite these differences, we should generally see similar methylation levels. If we compare methylation levels between methylation profiles from different library preparations, using a defined set of intervals, we would expect to see very little difference. The difference we do observe should reflect noise due to sampling in regions with low coverage. Because methylation data from two different libraries reflects different actual molecules, combining data from different libraries is easy. The methylation levels in `methcounts` files can be combined by simply merging the individual `methcounts` files. Similarly, the `methstates` files can be concatenated and the re-sorted. Within a biological replicate, we organize libraries as follows.

```
./ASDF/ASDF_R1/ASDF_R1_L1/
./ASDF/ASDF_R1/ASDF_R1_L2/
./ASDF/ASDF_R1/ASDF_R1_L3/
```

Reads files within a library For a given library, having a simple pipeline is facilitated by having a uniform scheme for naming the data files. The following example illustrates files corresponding to 4 sets of reads, two from paired-end sequencing and two from single-end sequencing:

```
./ASDF/ASDF_R1/ASDF_R1_L1/1_1.fq
./ASDF/ASDF_R1/ASDF_R1_L1/1_2.fq
./ASDF/ASDF_R1/ASDF_R1_L1/2_1.fq
./ASDF/ASDF_R1/ASDF_R1_L1/2_2.fq
./ASDF/ASDF_R1/ASDF_R1_L1/3.fq
./ASDF/ASDF_R1/ASDF_R1_L1/4.fq
```

The files `1_1.fq` and `1_2.fq` are corresponding left and right mates files. These must be processed together so that the corresponding mates can be joined. The file `3.fq` is from a single-end sequencing run. We recommend using symbolic links to set up these filenames, and to keep subdirectories corresponding to the individual sequencing runs for the same library.

```
./ASDF/ASDF_R1/ASDF_R1_L1/Run1/s_1_1_sequence.txt
./ASDF/ASDF_R1/ASDF_R1_L1/Run1/s_1_2_sequence.txt
./ASDF/ASDF_R1/ASDF_R1_L1/Run2/LID12345_NoIndex_L001_R1_001.fastq
./ASDF/ASDF_R1/ASDF_R1_L1/Run2/LID12345_NoIndex_L001_R2_001.fastq
./ASDF/ASDF_R1/ASDF_R1_L1/Run3/s_1_sequence.txt
./ASDF/ASDF_R1/ASDF_R1_L1/Run3/s_2_sequence.txt
./ASDF/ASDF_R1/ASDF_R1_L1/1_1.fq --> Run1/s_1_1_sequence.txt
./ASDF/ASDF_R1/ASDF_R1_L1/1_2.fq --> Run1/s_1_2_sequence.txt
./ASDF/ASDF_R1/ASDF_R1_L1/2_1.fq --> Run2/LID12345_NoIndex_L001_R1_001.fastq
./ASDF/ASDF_R1/ASDF_R1_L1/2_2.fq --> Run2/LID12345_NoIndex_L001_R2_001.fastq
./ASDF/ASDF_R1/ASDF_R1_L1/3.fq --> Run3/s_1_sequence.txt
./ASDF/ASDF_R1/ASDF_R1_L1/4.fq --> Run3/s_2_sequence.txt
```

This kind of organization accomplishes four things: (1) it keeps read files from different runs separate, (2) it provides a place to keep metadata for the different runs, (3) it allows the reads files to exist with their original names which might not be different between runs, and (4) it ensures a simple naming scheme for the data files needed by our pipeline.

The results directories For the example scheme we have been describing, we would organize the results in the following directories.

```
./ASDF/ASDF_R1/ASDF_R1_L1/results/
./ASDF/ASDF_R1/ASDF_R1_L2/results/
./ASDF/ASDF_R1/results/
./ASDF/ASDF_R2/ASDF_R2_L1/results/
./ASDF/ASDF_R2/ASDF_R2_L2/results/
./ASDF/ASDF_R2/results/
./ASDF/ASDF_R3/ASDF_R3_L1/results/
./ASDF/ASDF_R3/ASDF_R3_L2/results/
./ASDF/ASDF_R3/results/
./ASDF/results/
```

6 Auxiliary tools

6.1 Count number of lines in a big file

When working with next-generation sequencing data, you may usually needs to work with very big files, such as FASTQ files containing raw reads and `*.mr` files containing mapped reads. You may would like to know the approximate number of reads in those large files. The program `lc_approx` estimate the number of lines by counting the number of lines in a small chunk randomly chosen from the big file and scaling the estimate by file sizes. 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_apprix` can be hundreds of times faster than the unix tool `wc -l`.

6.2 Automating methylome analysis

Two bash scripts have been provided to perform quick and consistent analysis on the individual library level and aggregated biological replicate level. For each library in a project, `library` sorts, removes duplicate reads, calculates the bisulfite conversion rate, runs `methcounts`, and runs `levels`. There is also an option to perform `methcounts` analysis on all cytosines, which can be toggled by uncommenting the command in the script. For RRBS data, the `duplicate-remover` command should be commented.

To run `library`, navigate to the library's results directory – the structure should look something like

```
./ASDF/ASDF_R1/ASDF_R1_L1/results_mm9/
```

and should contain a single mapped reads file. To run, specify the directory where `methpipe` binaries can be found, the directory where chromosome files can be found, and the base name of the library or replicate you are working on:

```
library /home/user/Desktop/methpipe/trunk/bin \
        /home/user/Desktop/mm9_chroms ASDF_R1_L1
```

When `library` has been run on all individual libraries, the results can be merged to produce results for the biological replicate using `merge-methylomes`. This merges bisulfite conversion rate and methcounts statistics, and then uses them to generate HMRs, PMRs, and AMRs. To run `merge-methylomes`, navigate to the biological replicate's results directory – the structure should look something like

```
./ASDF/ASDF_R1/results_mm9/
```

and should be empty when the program starts. To run, specify the same command line arguments as above, with the biological replicate as the base name:

```
biorep /home/user/Desktop/methpipe/trunk/bin \
       /home/user/Desktop/mm9_chroms ASDF_R1 L
```

In some cases, it is useful to merge biological replicates rather than libraries. In this case, the fourth parameter of `merge-methylomes` should be `R`.

All run configurations for the programs in these scripts are consistent with data in methbase, and therefore direct comparison is appropriate. These tools should provide a convenient, consistent workflow for researchers to quickly analyze and compare their methylomes with those made publically available in methbase.