Methpipe Manual

Qiang Song Michael Kessler Fang Fang Jenny Qu Tyler Garvin
Andrew Smith

April 12, 2012

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.

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 familiary with BS-seq experiments, especially the bisulfite conversion reaction, and how this affects what we observe in the sequenced reads. Especially 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) consitution (A is the complimentary base pair of T), and are referred to as A-rich sequences. Mapping consists of finding sequence similary, based on context specific criteria, between these short sequences, or reads, and a homologous (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 approrpiately formatted for the next components of the `methpipe` pipeline (necessary file formats for each step are covered in the corresponding sections). The deault 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 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, name, score, 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`.

Partitioning reads for mapping using a cluster: Mapping reads often takes a while, and mapping reads from BS-seq takes longer. Because of how `rmapbs` works, dividing the set of reads to be mapped into k equal sized smaller reads files, and mapping these all simultaneously on a cluster, will make the mapping finish about k times faster. 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. 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 (example will be given below).

Sequencing adaptors: These are a problem in any sequencing experiment with short fragments relative to the lengths of reads. The `cutadapt` program does a fine job removing adaptors. If you plan to use `rmapbs` after removing the adaptors, then you will need to replace the parts of the reads that get removed with `N` characters. These will not influence the mapping, because they will be treated as wildcard characters by `rmapbs`, but reads are currently required to be the same length in `rmapbs`.

You can also have `rmapbs` cut the adaptors for you. All you need to do is provide the adaptor sequence on the command line. `rmapbs` will only cut the adaptor if it sees at least 10nt of the adaptor, and the rest of the adaptor sequence extending towards the 3' end of the read. This will work for both ends of paired-end reads. The situation of adaptor concatamers, where the adaptor begins immediately at the beginning of the read, may not be handled well.

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". We will follow this convention throught the manual and strongly suggest that you do the same. T-rich reads are sometimes referred to as $5'$ reads or mates 1 and A-rich reads are sometimes referred to $3'$ reads or mates 2; we will stick with T-rich and A-rich. `rmapbs` expects T-rich reads as a default and so for paired-end sequencing data, you must run `rmapbs` twice; once for T-rich reads and once for A-rich reads. Execute the following commands to map of your paired-end reads with `rmapbs`; one command for T-rich reads, and one for A-rich reads.

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

An example command for the second end (previously named like `s_1_2_sequence.txt` from Illumina), which will contain all A-rich reads:

```
$ ./rmapbs -c hg18 -o Human_ESC_2.mr -A Human_ESC_2.fastq
```

Merging paired-end reads: As previously noted, paired-end sequencing produces both T-rich reads and A-rich reads. Since methylation is estimated from the counts of Ts and Cs (that map to C's in the reference genome), and A-rich reads contain methylation information in the form of A and G counts (that map to Gs in the reference genome), A-rich reads must turned into T-rich reads by reverse complimenttion. This allows methylation information to be derived from the Ts and Cs in the new T-rich reads that correspond to the As and Gs from the A-rich reads that was reverse complementated.

A double counting error could arise if overlapping regions exist between any two mates. To avoid this error and maintain all other important information, we replace bases in the overlapping region with Ns in one of the two mates. Reverse-complemention of A-rich reads (and switching the strand to which they mapped), and masking overlapping regions, are done by the program `clipmates`. This tool works on mapped reads files that been sorted by read id (read name). The following command is an example of how to sort your mapped reads files with the Unix `sort` command before inputting them into `clipmates`.

```
$ sort -k 4,4 -o Human_ESC_1.mr.sorted Human_ESC_1.mr
$ sort -k 4,4 -o Human_ESC_2.mr.sorted Human_ESC_2.mr
```

The names of corresponding mates must only differ by the last character (which should be a 1 for T-rich reads and 2 for A-rich reads). If mates exist and are mapped correctly (to the same chromosome, to correct strands, with correct orientation) within a certain distance from each other (as specified by the `-L` option), then the mates are combined into a single read, referred to as a fragment, with Ns filling any existing gaps between mates. If mates could not be matched, then each mate is present in the output file. `clipmates` takes two input files in mapped reads format: one with T-rich mreads (`-T` option) and the other with A-rich reads (`-A` option). Execute the following command to run the `clipmates` component of `methpipe` on T-rich and A-rich reads (mates 1 and 2).

```
$ ./clipmates -S Human_ESC.clipstats -o Human_ESC.mr \
              -T Human_ESC_1.mr.name_sorted -A Human_ESC_2.mr.name_sorted
```

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` can be used to remove such duplicates. It collects reads and fragments mapped to the same genomic location (start or end), and chooses a random one to be the representative of the original DNA sequence. Since sequencing can run for variable lengths of time (leading to different length reads), we must sort reads by the position of the side where sequenceing starts in order to find duplicates. This is always the 5' side, which for + strand reads is their start positions and for - strand reads is the end positions. Therefore, two separte sorts and removals of duplicates must be done; one on + strand reads while sorting on start positions, and one on - strand reads while sorting on end position. The `-B` option of `duplicate-remover` ignores + reads and tells the program that reads are sorted by end position. To remove read duplicates, follow the following steps (examples are provided below): (1) sort reads by start position (chrom, start, end, strand), (2) merge files with reads that came from the same DNA library (and therefore underwent the same PCR) (3) run `duplicate-remover` to remove read duplicates mapping to the same start positions, (4) sort reads by end position (chrom, end, start, strand), and (5) run `duplicate-remover` with option `-B` to remove duplicates mapping to the same end positions.

First, sort reads by start position (chrom, start, end, strand). Remember, for paired-end reads, use the output from climpates while working through the following steps:

```
$ sort -k 1,1 -k 2,2g -k 3,3g -k 6,6 \
       -o Human_ESC.mr.sorted_start Human_ESC.mr
$ sort -k 1,1 -k 2,2g -k 3,3g -k 6,6 \
       -o Human_NHFF.mr.sorted_start Human_NHFF.mr
```

If at any point during the pipeline you split your files into smaller files due to size limitations, you should now merge all these split reads files back into one complete file. We recommend that you do this using `-m` option of the Unix command `sort`. The following command is an example of how to merge three such files with the assumption that these files were labled with a 1, 2 or 3 respectively.

```
$ sort -m -k 1,1 -k 2,2g -k 6,6 -k 3,3g \
       -o Human_ESC.mr.sorted_start_merged \
       Human_ESC_1.mr.sorted_start \
       Human_ESC_2.mr.sorted_start \
       Human_ESC_3.mr.sorted_start
$ sort -m -k 1,1 -k 2,2g -k 6,6 -k 3,3g \
       -o Human_NHFF.mr.sorted_start_merged \
       Human_NHFF_1.mr.sorted_start \
       Human_NHFF_2.mr.sorted_start \
       Human_NHFF_3.mr.sorted_start
```

Execute the following command to remove read duplicates sharing the same start position by running the `duplicate-remover` component of `methpipe`:

```
$ ./duplicate-remover -S Human_ESC_dremove_start_stat.txt \
                      -o Human_ESC_dremove_start.mr Human_ESC_sorted_start.mr
$ ./duplicate-remover -S Human_NHFF_dremove_start_stat.txt \
                      -o Human_NHFF_dremove_start.mr Human_NHFF_sorted_start.mr
```

Next, sort reads by end position: chrom, end, start, strand.

```
$ sort -k 1,1 -k 3,3g -k 2,2g -k 6,6 \
       -o Human_ESC.mr.sorted_end Human_ESC.mr
$ sort -k 1,1 -k 3,3g -k 2,2g -k 6,6 \
       -o Human_NHFF.mr.sorted_end Human_NHFF.mr
```

Execute the following command to remove read duplicates sharing the same end position by running the `duplicate-remover` component of Methpipe.

```
$ ./duplicate-remover -S Human_ESC_dremove_start_stat.txt \
                      -o Human_ESC_dremove_end.mr -B Human_ESC_sorted_end.mr
$ ./duplicate-remover -S Human_NHFF_dremove_start_stat.txt \
                      -o Human_NHFF_dremove_end.mr -B Human_NHFF_sorted_end.mr
```

Need to describe the output format for the stats on the number of distinct reads. Also need to describe the reorder tool so that re-sorting is not so painful.

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., 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 Cs 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
```

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 output is in 6-column BED format. `methcounts` requires that the input reads are sorted according to (chrom, end, start, strand). If your reads are not sorted, to sort MappedRead format files in this order, do:

```
$ sort -k 1,1 -k 3,3g -k 2,2g -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:

```
$ sort -m -k 1,1 -k 3,3g -k 2,2g -k 6,6 \
       -o Human_ESC.mr.sorted_end_first Human_ESC.mr.1 Human_ESC.mr.2
```

Counting only CpG sites: Methylation is estimated at single base resolution for every cytosine loci and is represented as a probability based on the ratio of methylated to unmethylated reads mapping to that loci. To produce methylation levels at each CpG site you can use commands like the following:

```
$ ./methcounts -c hg18 -o Human_ESC_Meth.bed \
               -S Human_ESC_Meth.stats Human_ESC.mr
$ ./methcounts -c hg18 -o Human_NHFF_Meth.bed \
               -S Human_NHFF_Meth.stats Human_NHFF.mr
```

The output will contain one line per CpG site, and to conform to BED format we indicate CpG sites as genomic intervals of width 1. The first column is the chromosome, the second is the location of the CpG site, the 3rd column is equal to the 2nd + 1. The 4th column is the "name" and includes 2 parts: a tag indicating the type of site (in this case it will be "CpG") and an integer indicating the number of reads mapping over the site (in this case on either strand) that has either a C or a T at that position. The two parts are separated by a colon (:). Sequencing errors are not counted in the output. 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 the strand, and when only CpG sites are considered, this is always +.

Counting all cytosines: In mammals, DNA methylation exists mostly at cytosines in the context of CpG dinucleotides. This kind of methylation is always symmetric due to the interesting action of DNA methyltransferases, which methylate GpC dinucleotides on the complemtary strand of methylated CpG dinucleotides during DNA replication. In mammalian stem cells and in 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 often referred to as asymmetric since the cytosines on the complementary strand are not necessarily methylated. 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, use the `-N` option. The following command calculates methylation using the `methcounts` component of methpipe.

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), differentially methylated regions (DMRs), and regions with allele-specific methylation (ASM).

3.1 Hypomethylated and hypermethylated regions (HMRs)

Methylomes almost always have bimodal methylation distributions, with one peak representing higher methylation levels and the other representing lower methylation levels. Statistical approaches are employed to estimate these peaks, calculate certain parameters, such as the average distance between a highly methylated site and its closest lowly methylated site, and ultimately determine probabilistically what constitutes high, intermediate, or low levels of methylation. Criteria for such classifications are based on the characteristiscs of the specific data set (i.e. methylome) being analyzed, and interesting genome-wide methylation patterns can often be identified. One such pattern is the occurence of hypomethylated regions (HMRs), or contiguous regions with low methylation levels. Methylation is often a mechanism by which gene expression is significantly decreased, and so regions of decreased methylation may

signify genetically important regions with high levels of gene expression. To identify HMRs, execute the following command that runs the program `hmr` on the output of `methcounts`.

```
$ ./hmr -o Human_ESC_HMR.bed Human_ESC_Meth.bed
$ ./hmr -o Human_NHFF_HMR.bed Human_NHFF_Meth.bed
```

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 accross two methylomes using the `methdiff` component of Methpipe.

```
$ ./methdiff -o Human_ESC_NHFF_methdiff.bed \
             Human_ESC_Meth.bed Human_NHFF_Meth.bed
```

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. DMR's are regions where differential methylation scores indicate there are many CpGs with a high probability of being differentially methylated between the two methylomes. Option `-l` regulates the output: without this option, DMRs are calculated, in which methylation is less in one methylome, and with this option DMRs are calculated with methylation less in the other methylome. The following command finds DMRs using the `dmr` component of `methpipe`:

```
$ ./dmr -o Human_ESC_NHFF_DMR.bed -d 500 -b 10 -C 10 -m 200
        -c 0.7 Human_ESC_NHFF_methdiff.bed
```

3.3 Allele-specific methylation

The program `allelicmeth` calculates allelic-specific methylation scores for each CpG site. The higher the score, the more likely the site has allelic-specific methylation. The allelic scores profile can be visualized using a genome browser. Input files should be the mapped reads files (`.mr` suffix) produced previously in the mapping step. Each CpG in the output file represents the CpG pairs made by any CpG and its two adjacent CpGs. Every read mapping over either CpG pair is included in the number of mapped reads in the output file and the number of reads with methylation states for the CpG pair of either, methylated methylated (mm), methylated unmethylated (mu), unmethylated methylated (um), or unmethylated unmethylated (uu) is tallied. **NOTE TO SELF: this needs ot be addressed further and made more clear. As it is it is wrong. I just wanted to get something on paper before forgetting it.** The following command will calculate allelic methylation scores using the **allelicmeth** component of Methpipe.

```
$ ./allelicmeth -c hg18 -o Human_ESC_allelicmeth.bed Human_ESC.mr
$ ./allelicmeth -c hg18 -o Human_NHFF_allelicmeth.bed Human_NHFF.mr
```

3.3.1 Allelically methylated regions (AMRs)

In diploid organisms like human, the two sets of chromosomes may contain different methylation states at some regions (*e.g.* imprinting control regions), which are called 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 following command shows an example to run the program `amrfinder`.

```
$ ./amrfinder -o Human_ESC_AMR.bed -i 10 -w 10 -m 1 -c hg18 Human_ESC.mr
```

Option `-o` indicates the output bed file, which contain all possible AMRs in BED format. 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. Option `-m` is the requirement of the minimum reads covering each CpG, and the default value is 1. 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 numering 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 | cut -f 1,2,7 > Human_ESC.txt
```

3.3.2 Optimizing boundaries of AMRs

To optimize boundaries of AMRs based on the initial genome-wide scan results, the program `amrrefiner` uses a dynamic programming method to test all ways to partition a genomic interval into alternating subintervals of AMRs and non-AMRs. This program provides more precise prediction but is much more computationally expensive. Therefore, the program is not appropriate for genome-wide application, but only suitable for applying in genomic intervals where you have prior information of the existence of AMRs. For example, we know that the genomic interval around the human GNAS gene should have AMRs. The CpGs in this interval are extracted into a bed file `GNAS_cpg.bed`. Then the AMRs around GNAS can be located as below.

```
$ ./amrrefiner -o Human_ESC_GNAS_AMR.bed -i 10 -s 10 -m 100 -M 400 \
               -e 1 -c hg18 -E GNAS_cpg.bed Human_ESC.txt
```

Options `-i` and `-E` are the same as the program `amrfinder`. Option `-s` is the minimum AMR size, `-m` is the mean AMR size and `-M` is the maximum AMR size. All these three options are in terms of number of CpGs and help to estimate the size of AMR. Option `-e` is the expected number of AMRs in the interrogated genomic interval. The input read file `Human_ESC.txt` is in the epiread format mentioned above.

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 `methcounts` output file and a BED format file of genomic "regions of interest" (hence the "roi" in `roimethstat`). It can be run as follows:

```
$ ./roimethstat -o regions_ESC_meth.bed -r regions.bed Human_ESC_Meth.bed
```

The output format is also 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.

4 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.bed` file, which is the output of `methcount` program. A methcount 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. hg18). Note that this is the file that is referred to as `hg18.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 hg18.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 hg18.chrom.sizes Human_ESC_Reads.bw
   ```

You might also want to create `bidBed` 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. hg18). Note that this is the file that is referred to as `hg18.chrom.sizes` in step 3.

3. Modify and use the following comands: 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), $6}' \
         Human_ESC_HMR.bed > Human_ESC_HMR.bed.tmp
   $ bedToBigBed Human_ESC_HMR.bed.tmp hg18.chrom.sizes Human_ESC_HMR.bb
   ```

   Otherwise, one can run the following command directly.

   ```
   $ bedToBigBed Human_ESC_HMR.bed hg18.chrom.sizes Human_ESC_HMR.bb
   ```