Methpipe Manual

November 4, 2010

The methpipe software package is a comprehensive tool chain for analyzing whole genome bisulfite sequencing data (BS-Seq). This documentation will guide you step by step to learn how to perform the data analysis in a BS-Seq project with methpipe. To facilitate understanding of the work flow, we divide the analysis procedure into four steps: (1) Pre-mapping processing: this step include assessing read quality and pre-processing reads, such as trimming adapters; (2) Mapping: this step maps sequence reads to a reference genome; (3) Analyzing methylation status at single site: in this step, we will estimate methylation frequency at single site, either CpG sites or non-CpG sites. We will also obtain information about sequencing depth and bisulfite conversion rate; (4) Higher level analysis: this step includes higher level more biologically interesting analysis, such as identifying hypo-methylated regions and/or differentially methylated regions. Later we will describe in detail the usage of tools in each step.

In the guide we will use a small project as example. Suppose that in this project you would like to study the methylation pattern in two cell types, B cells and neutrophiles. The B cell methylome is profiled with sing-end sequencing and your university's sequencing center sends back to you these sequence files:

```
bcell/s-1.fq bcell/s-2.fq bcell/s-3.fq bcell/s-4.fq.
```

The neutrophile methylome is profiled with pair-end sequencing and you have the following sequences files:

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

Now we are ready to uncover the interesting biology about DNA methylation from this dataset.

1 Pre-mapping processing

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

1.1 Trim adapters

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

The program **trim-adapter** is used to trim adapter sequences, if any, from the 3' end of raw reads. Suppose the adapter sequences is *GATCGGAAGAGCGGTTCAGCAGGAATGCCGAGACCGATCTCGTATGCCG*, we trim the adapter sequences from raw reads in the file bcell/s-1.fq with the following command:

```
$ ./trim-adapter  s-1.fq -o s-1-clipped.fq
-a GATCGGAAGAGCGGTTCAGCAGGAATGCCGAGACCGATCTCGTATGCCG
```

If you would like to check the effective read length after trimming adapter sequences and Ns from 3' end, you can add **-l** options to specify the output file for effective read length, for example,

```
$ ./trim-adapter  s-1.fq -o s-1-clipped.fq  -l s-1-length.txt
-a GATCGGAAGAGCGGTTCAGCAGGAATGCCGAGACCGATCTCGTATGCCG
```

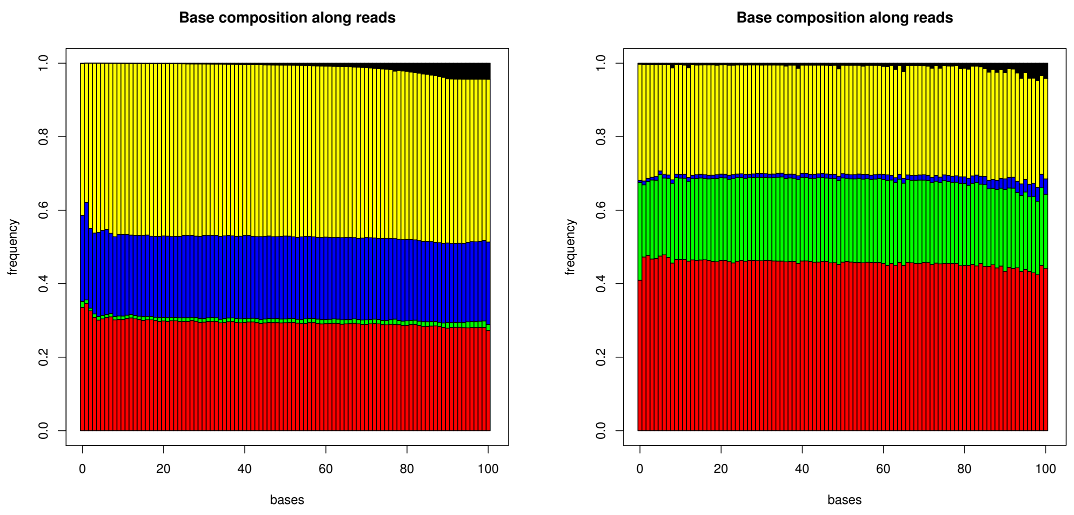
The file s-1-length.txt lists the effective read length of each read after trimming adapter sequences and Ns, which can be used to compute the statistics of effective reads lengths with any general purpose statistical package, such as R. Perform this trimming adapter processing on all sequences files and for convenience we assume the processed sequences files have their original file names thereafter.

1.2 Assessing read quality

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

```
$ ./read-quality-prof  < s-1.fq > s-1-qual.txt
```

The output file s-1-qual.txt can be visualized with the R software. Fig. 1 shows the base composition profile in the pair-end sequencing sample, where the left panel shows T-rich reads and the right panel A-rich reads. Note the small proportion of C and G reflects the effect of bisulfite conversion.



(a) T-rich reads (b) A-rich reads

Figure 1: Base composition

2 Mapping

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

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

2.1 Mapping single-end sequencing reads

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

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

Here we briefly explain some commonly used options of **rmapbs**. The option **-c** specifies the direcotory holding the reference genome and the **-s** specifies the suffix of sequence file names. The option **-m** gives the maximum number of mismatches allowed. The option **-o** specifies the output file.

2.2 Mapping pair-end sequencing reads

2.3 Parallel mapping

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

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

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

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

3 Post-mapping processing

merge.cpp merge sorted MappedRead file, with option to control whether remove redundant file

mask-overlap.cpp mask overlapping region of paired-reads, also generate summary of fragment length, number of unpaired reads

unique.cpp filter program to remove duplicate reads

sort.cpp sort MappedRead file: either by genomic location or by name

revcomp.cpp do reverse complement operation on MappedRead

4 Analysis

methcount.cpp This program reads mapped read file and output methylation frequency for each CpG site.

bsrate.cpp This program reads mapped read file and estimate bisulfite conversion rate by checking methylation status of non-CpG C's

5 Visualization

6 A sample work flow

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

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

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

```
# Pre-mapping processing #

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

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

# post-mapping processing

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

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

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

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

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

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

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