Genomewide methylation heterogeneity and differential heterogeneity analysis

Contents

1 Overview

The overview of the analysis of methylation heterogeneity can be found in Figure 1. The analysis is divided into two main steps: genome screening and differential heterogeneity analysis. Genome screening includes obtaining information necessary for the estimation of methylation heterogeneity (MeH) from its raw data. It extracts bisulfite converted reads and the genomic locations they're mapped to to summarise the compositions of methylation patterns within windows of multiple cytosines (of a given context; i.e., CG). It then evaluates MeH given the window has enough reads and outputs the results (MeH and the starting CpG position) as a row in an excel file. This process is done within a python program called genome_scr.py. The program not only evaluates MeH genome-wide but also preprocess (split the bam files for multiprocessing) for computational efficiency and merge results from several files (samples) after genome screening. After the results from several samples are merged, differential heterogeneous regions can be identified using DHR.R.

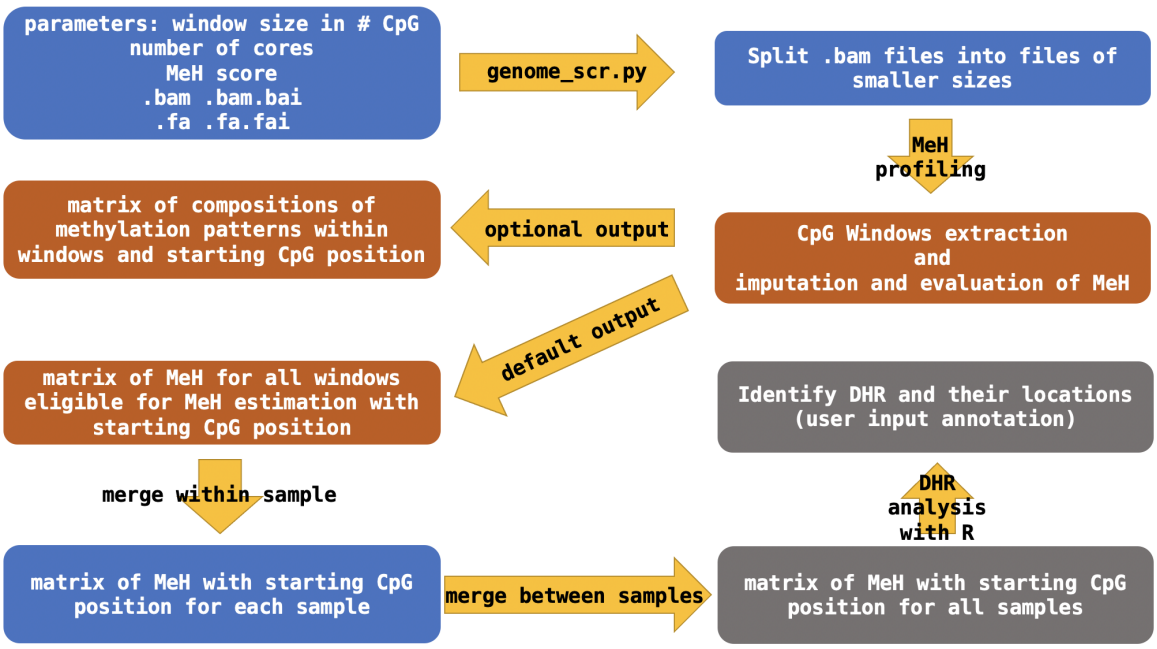

Figure 1: Overview of using the program to analyse MeH.

2 Methylation heterogeneity profiling (MeH profiling)

Genome screening is the process of obtaining information necessary for the estimation of methylation heterogeneity, estimating methylation heterogeneity and recording the results. It is done by sliding windows of 1 cytosine and recording the bisulfite converted reads mapped to the genomic location. Once enough cytosines are gone through, they are imputed if eligible and evaluated for MeH if there are enough reads. There are a few choices offered for users. They are window size w or the number of cytosines for the evaluation of 1 methylation heterogeneity score, the default is 4, minimum number of reads for the evaluation of MeH, the default is 2^{w-2}, number of cores c to be used for multiprocessing, the default value is 4, what methylation contexts to consider, the default is CG sites, and whether the compositions of methylation patterns will be outputted, i.e., CG, CHG or CHH (or any combination of these). Since information such as methylation patterns including methylation statuses and genomic position mapped to for each base pair of every read is required, the program needs both the original alignment files and the reference genome, that are given by ".bam" and ".fa". In order to obtain ".bam", a standard workflow is also suggested in Figure 2.

2.1 Obtaining inputs for genome screening

Before MeH can be profiled, users need to obtain .bam files, that is, mapped bisulfite converted reads. The standard workflow is summarised in Figure 2.

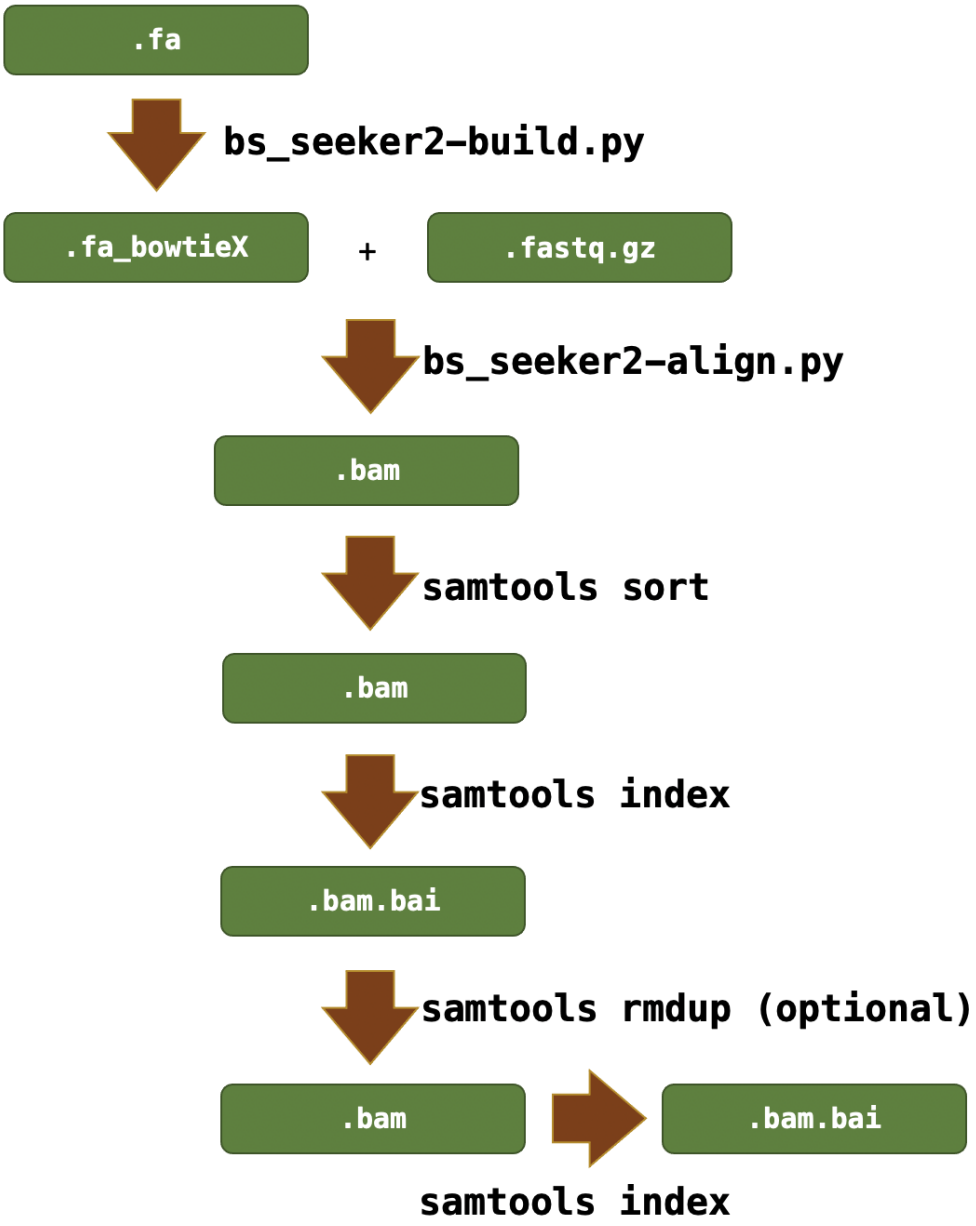

Figure 2: Preprocessing of data before inputting into the program.

Once all the bam files (and their corresponding index files) are obtained, they need to be placed in a folder named "MeHdata" together with the reference genome and its index file.

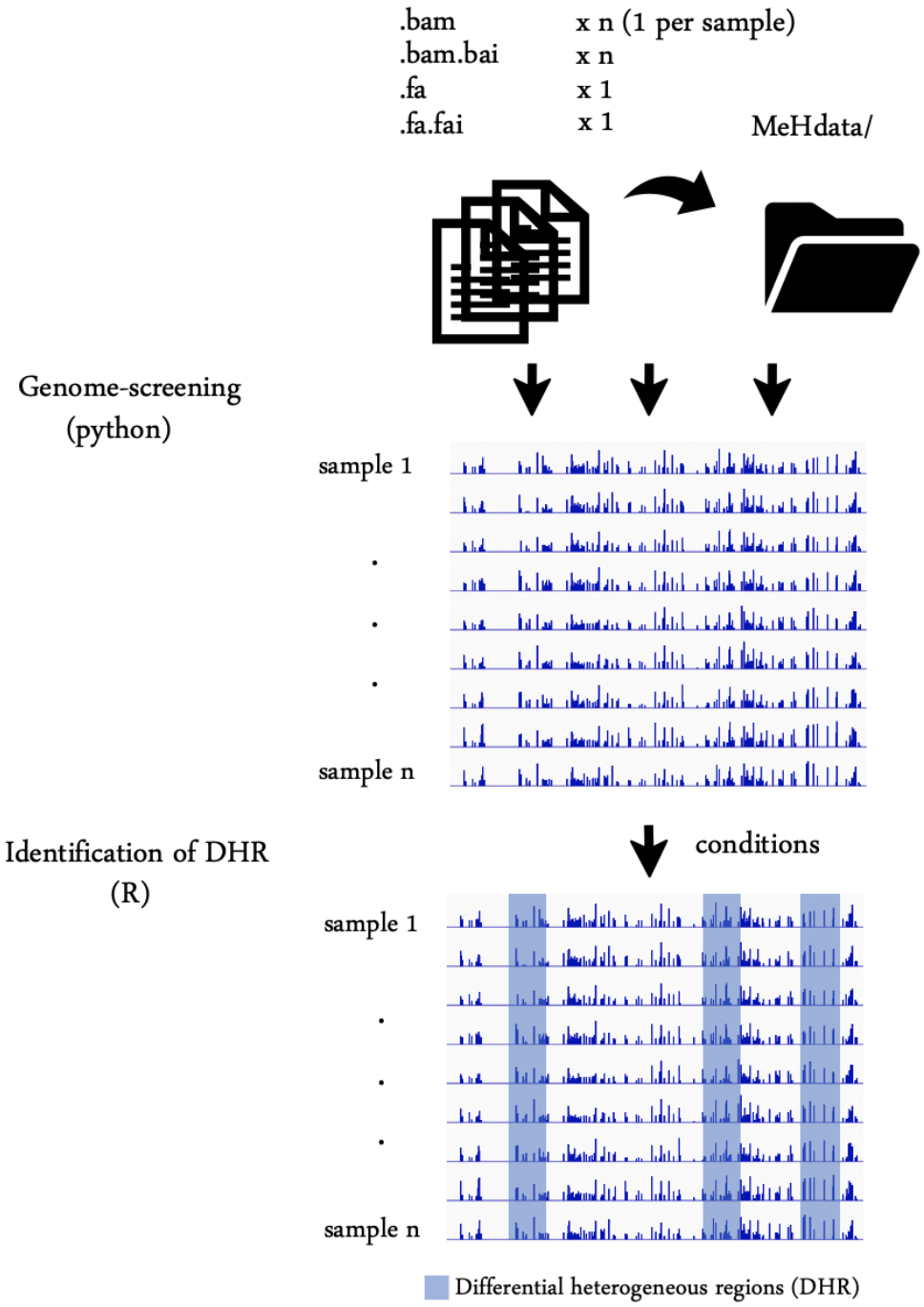

Figure 3: Visualisation of the process.

2.2 Parameters for genome screening

- -w, window size in numbers of cytosines, default is 4

- -m, scores for evaluating methylation heterogeneity, default is 2: pairwise-similarity-based method, other options include 1: Phylogeny-based method, 2: Abundance-based method, 4: Entropy and 5: Epipolymorphism.
- -d, distance used to calculate pairwise distances between methylation patterns, default is 1: Hamming distance, other option include 2: weighted degree kernel. Only required for Phylogeny-based method and pairwise-similarity-based method.
- -c, number of cores for parallel processing, default is 4
- –CG, to evaluate context 'CG', default is False
- –CHG, to evaluate context 'CHG', default is False
- –CHH, to evaluate context 'CHH', default is False
- –opt, to output statistics of methylation patterns (how many copies of each methylation pattern at all windows eligible for the evaluation of MeH), default is False
- –mlv, to output methylation level for all cytosines under the genomic contexts specified as long as the depth is greater than 3, results will be merged into bins, default is False

2.3 Output from genome screening

2.3.1 Default output

When the program genome_scr.py is run, all bam files in the folder will be processed and results from all samples will be merged into one .csv file containing the MeH of each sample and their genomic locations.

	A	B	C	D	E	F
1	chrom	bin	strand	testAT31ch12	C0035test1234	C0037test1234
2	1	600	r	5.20078	6.997035	3.87375
3	1	3800	f	0.377122667	0.40406	0.094280667
4	1	4200	f	0	1.0606575	0.707105
5	1	4600	r	0	0	0
6	1	5000	f	0	0.774559	0
7	1	5000	r	0	0.628537778	0
8	1	5400	r	0	0.60609	0

Figure 4: Visualisation of the default output.

2.3.2 Optional output

The intermediate product, that is, the compositions of methylation patterns for each window can be outputted if user chooses to. It will also be in .csv with each row containing the compositions of a window and its starting (cytosine) genomic location.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
1	chrom	pos	p01	p02	p03	p04	p05	p06	p07	p08	p09	p10	p11	p12	p13	p14	p15	p16	MeH	dis	ML	depth	strand
2	1	510	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	25	2.07834	139	0.767	30	f
3	1	641	0	0	0	0	0	0	0	8	0	0	0	0	0	1	1	18	2.05722	148	0.963	27	f
4	1	790	0	0	0	0	0	0	0	4	0	0	0	0	0	2	3	13	3.25572	114	0.913	23	r
5	1	809	0	0	0	4	0	0	0	2	0	0	0	2	0	0	1	16	3.44206	102	0.926	27	r
6	1	839	0	5	0	3	0	1	0	11	0	0	0	0	0	1	0	9	4.76276	109	1	32	r
7	1	2296	39	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	141	0	20	r
8	1	2400	41	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	78	0	48	r
9	1	2431	27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	78	0	44	r

Figure 5: Visualisation of the optional output.

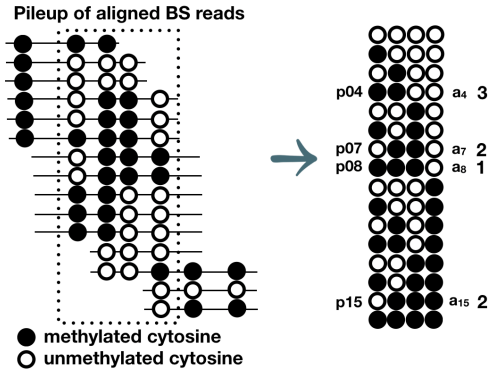

Figure 6: Explanation of how to obtain the abundances.

3 Differential heterogeneous (methylation) regions (DHR) analysis

DHR analysis is done in R with example scripts given as an .R file. The users need to load the output into R themselves.

3.1 Outputs of genome screening

The outputs from genome_scr.py contain mean MeH calculated using all windows eligible for MeH estimation within bins of 400bps. Each row represents a common bin with MeH value for all samples provided as bam files.

3.2 DHR Identification

The users need to give conditions to each sample; i.e., wildtype and mutant or stages 1 to 3 ...etc, and comparisons can be done for user specified conditions; i.e., stage 3 versus stage 1 or stage 2 versus stage 1. The output from this comparison will be a matrix consisting of difference in mean MeH between the conditions specified, p-values given by t-test, mean MeH of samples from the second condition, and mean MeH of samples from the first condition.

3.2.1 Statistical tests

Student's t-test is used for the comparison between conditions specified by the user under the null hypothesis of there is no difference between the mean MeH of the conditions specified and alternative hypothesis of the difference in the mean MeH of the conditions is nonzero. After the comparisons are done, the users can select regions (bins) based on p-value and delta (difference in mean MeH between the conditions specified) cutoffs of interest.

3.2.2 Annotation to input

If user chooses to provide annotation with the name of the genes, chromosome, transcription starting position and ending position then a list of differentially heterogeneous genes can be selected based on criteria given (how many bp upstream and downstream from TSS).

```
install.packages("roperators")
library(roperators)
install.packages("dplyr")
library(dplyr)
install.packages("foreach")
library(foreach)

MeH.t = function(vector,conditions,compare) {
ind1<-which(conditions == compare[1])+3 # +3 for chrom,bin and strand columns
ind2<-which(conditions == compare[2])+3
vector=as.data.frame(vector)
mean2=mean(as.numeric(vector[ind2]),na.rm=TRUE)
mean1=mean(as.numeric(vector[ind1]),na.rm=TRUE)
diff=mean2-mean1
if(sd(vector[ind1])<1e-5 && sd(vector[ind2])<1e-5)
return(data.frame(chrom=vector[1],pos=vector[2],delta=diff,pvalue=NaN,
mean2=mean2,mean1=mean1))
else {
out=t.test(vector[ind1],vector[ind2])
return(data.frame(chrom=vector[1],pos=vector[2],delta=out$est[2]-out$est[1],
pvalue=as.numeric(out$p.value),mean2=out$est[2],mean1=out$est[1]))
}
}



findgene = function(position) {
chr=as.character(position[1])
BP=as.numeric(position[2])
St=as.character(position[3])
Gene=geneloc$gene[which((geneloc$TSS<=BP)*(geneloc$TES>=BP)*
(as.character(geneloc$chrom)==chr)*(as.character(geneloc$strand)==
as.character(St))==1)][1]
if (St=='f') {
promoter=geneloc$gene[which((geneloc$TSS-1000<=BP)*(geneloc$TSS+1000>=BP)*
(as.character(geneloc$chrom)==chr)*(geneloc$strand=="f")==1)][1]
}
```

```
if (St=='r') {
promoter=geneloc$gene[which((geneloc$TES-1000<=BP)*(geneloc$TES+1000>=BP)*
(as.character(geneloc$chrom)==chr)*(geneloc$strand=="r")==1)][1]
}
return(list(chrom=chr,bin=BP,Gene=Gene,Promoter=promoter,strand=St))
}

#### Load files for analysis by first setting the work directory
#### to where your files are located
setwd("~/MeHdata")
CG <- read.table('CG_Results.csv',header=TRUE,sep=",")
CHG <- read.table('CHG_Results.csv',header=TRUE,sep=",")
CHH <- read.table('CHH_Results.csv',header=TRUE,sep=",")

#### Define conditions of all samples; i.e., A and B for 2 conditions,
#### each with two replicates, samples 1 and 2 are replicates of A and
#### samples 3 and 4 are replicates for B. This is for comparisons to be
#### carried out later on
conditions <- c("A","A","B","B")

#### Calculate t-statistics and p-values for all bins between
#### user specified conditions
install.packages("roperators")
library(roperators)
install.packages("dplyr")
library(dplyr)
install.packages("foreach")
library(foreach)
library(doParallel)
registerDoParallel(cores=4)

#### Compare condition B with A (baseline)
Comp1<-data.frame(foreach(i = 1:dim(CG)[1],.combine = rbind) %dopar%
MeH.t(CG[i,],conditions=conditions,c("A","B")))
Comp1$padj=p.adjust(Comp1$pvalue)
```

```
#### Select differential heterogeneous regions based on user
#### specified conditions; i.e., p-value of 0.05 and delta of 1 (positive or negative)
Comp1$DHR <- (Comp1$padj<0.05)*(abs(Comp1$delta)>1.4)
Comp1$DHR <- (Comp1$pvalue<0.05)*(abs(Comp1$delta)>1.4)
Comp1$DHR.up <- (Comp1$pvalue<0.05)*(Comp1$delta>1.4)
Comp1$DHR.down <- (Comp1$pvalue<0.05)*(Comp1$delta<(-1.4))

#### DHG analysis if bed file is given as .txt with each row representing
#### a gene and consists of gene name, chromosome number, TSS, TES and
#### strand as 'f' (forward) or 'r' (reverse)
geneloc<-read.table('genelist.txt',header=TRUE)
colnames(geneloc)<-c("gene","chrom","strand","TSS","TES")
geneloc$strand[as.character(geneloc$strand)=="+"]<-"f"
geneloc$strand[as.character(geneloc$strand)=="-"]<-"r"

genelist<-as.data.frame(foreach(i = 1:dim(Comp1)[1],.combine = rbind)
%dopar% findgene(Comp1[i,c("chrom","bin","strand")]))
```