MLML2R package User's Guide

Samara F. Kiihl and Maria Tellez-Plaza

2019-04-06

Abstract

We present a guide to the *R* package *MLML2R*. The package provides computationally efficient maximum likelihood estimates of DNA methylation and hydroxymethylation proportions when single nucleotide resolution data from the DNA processing methods bisulfite conversion (BS), oxidative bisulfite conversion (ox-BS), and Tet-assisted bisulfite conversion (TAB) are available. Estimates can be obtained by combining any two of the methods, or all the three methods (if available). The package does not depend on other *R* packages, allowing the user to read and preprocess the data with any given software, to import the results into *R* in matrix format, to obtain the maximum likelihood estimates for methylation and hydroxymethylation proportions and use them as input for other packages traditionally used in genomic data analysis, such as *minfi*, *sva* and *limma*.

Package

MLML2R 0.3.2

Contents

1 Introduction

In a given CpG site from a single cell we will either have a C or a T after bisulfite-based DNA conversion methods. We asume a Binomial model and maximum likelihood estimation to obtain consistent hydroxymethylation and methylation proportions with single nucleotide resolution. *MLML2R* package allows the user to jointly estimate hydroxymethylation and methylation consistently and efficiently.

T reads are referred to as converted cytosine and C reads are referred to as unconverted cytosine. In case of Infinium Methylation arrays, we have intensities representing the unconverted (M) and converted (U) channels. The most used summary from these experiments is the proportion $\beta = \frac{M}{M+U}$, commonly referred to as *beta-value*. Naively using the difference between betas from BS and oxBS as an estimate of 5-hmC (hydroxymethylated cytosine), and the difference between betas from BS and TAB as an estimate of 5-mC (methylated cytosine) can many times provide negative proportions and instances where the sum of uC (unmodified cytosine), 5-mC and 5-hmC proportions is greater than one due.

The function `MLML` takes as input the data from the different bisulfite-based methods and returns the estimated proportion of methylation, hydroxymethylation and unmethylation for a given CpG site. Table 1 presents the arguments of the `MLML` and Table 2 lists the results returned by the function.

The function assumes that the order of the samples by rows and columns in the input matrices is consistent. In addition, all the input matrices must have the same dimension. In the provided examples, rows represent CpG loci and columns represent samples. Nonetheless transposed matrices can also be supplied.

Table 1: `MLML` function and random variable notation

Arguments	Description
`U.matrix`	Converted cytosines (T counts or U channel) from standard BS-conversion (reflecting True 5-C).
`T.matrix`	Unconverted cytosines (C counts or M channel) from standard BS-conversion (reflecting 5-mC+5-hmC).
`G.matrix`	Converted cytosines (T counts or U channel) from TAB-conversion (reflecting 5-C + 5-mC).
`H.matrix`	Unconverted cytosines (C counts or M channel) from TAB-conversion (reflecting True 5-hmC).
`L.matrix`	Converted cytosines (T counts or U channel) from oxBS-conversion (reflecting 5-C + 5-hmC).
`M.matrix`	Unconverted cytosines (C counts or M channel) from oxBS-conversion (reflecting True 5-mC).

Table 2: Results returned from the `MLML` function

Value	Description
`mC`	maximum likelihood estimate for the proportion of methylation
`hmC`	maximum likelihood estimate for the proportion of hydroxymethylation
`C`	maximum likelihood estimate for the proportion of unmethylation
`methods`	the conversion methods used to produce the MLE

2 Worked examples

2.1 Publicly available array data: oxBS and BS methods

We will use the dataset from Field (2015), which consists of eight DNA samples from the same DNA source treated with oxBS and BS and hybridized to the Infinium 450K array.

When data is obtained through Infinium Methylation arrays, we recommend the use of the *minfi* package (Aryee et al. 2014), a well-established tool for reading, preprocessing and analysing DNA methylation data from these platforms. Although our example relies on *minfi* and other *Bioconductor* tools, *MLML2R* does not depend on any packages. Thus, the user is free to read and preprocess the data using any software of preference and then import into *R* in matrix format the intensities from the M and U channels (or C and T counts from sequencing) reflecting unconverted and converted cytosines, respectively.

To start this example we will need the following packages:

```
library(MLML2R)
library(minfi)
library(GEOquery)
library(IlluminaHumanMethylation450kmanifest)
```

It is usually best practice to start the analysis from the raw data, which in the case of the 450K array is a `.IDAT` file.

The raw files are deposited in GEO and can be downloaded by using the `getGEOSuppFiles`. There are two files for each replicate, since the 450k array is a two-color array. The `.IDAT` files are downloaded in compressed format and need to be uncompressed before they are read by the `read.metharray.exp` function.

```
getGEOSuppFiles("GSE63179")
untar("GSE63179/GSE63179_RAW.tar", exdir = "GSE63179/idat")

list.files("GSE63179/idat", pattern = "idat")
files <- list.files("GSE63179/idat", pattern = "idat.gz$", full = TRUE)
sapply(files, gunzip, overwrite = TRUE)
```

The `.IDAT` files can now be read:

```
rgSet <- read.metharray.exp("GSE63179/idat")
```

To access phenotype data we use the `pData` function. The phenotype data is not yet available from the `rgSet`.

```
pData(rgSet)
```

In this example the phenotype is not really relevant, since we have only one sample: male, 25 years old. What we do need is the information about the conversion method used in each replicate: BS or oxBS. We will access this information automatically from GEO:

```
if (!file.exists("GSE63179/GSE63179_series_matrix.txt.gz"))
download.file(
"https://ftp.ncbi.nlm.nih.gov/geo/series/GSE63nnn/GSE63179/matrix/GSE63179_series_matrix.txt.gz",
```

```
        "GSE63179/GSE63179_series_matrix.txt.gz")

        geoMat <- getGEO(filename="GSE63179/GSE63179_series_matrix.txt.gz",getGPL=FALSE)
        pD.all <- pData(geoMat)

        #Another option
        #geoMat <- getGEO("GSE63179")
        #pD.all <- pData(geoMat[[1]])

        pD <- pD.all[, c("title", "geo_accession", "characteristics_ch1.1",
                         "characteristics_ch1.2","characteristics_ch1.3")]
        pD
```

This phenotype data needs to be merged into the methylation data. The following commands guarantee we have the same replicate identifier in both datasets before merging.

```
sampleNames(rgSet) <- sapply(sampleNames(rgSet),function(x)
  strsplit(x,"_")[[1]][1])
rownames(pD) <- pD$geo_accession
pD <- pD[sampleNames(rgSet),]
pData(rgSet) <- as(pD,"DataFrame")
rgSet
```

The `rgSet` is an object from *RGChannelSet* class used for two color data (green and red channels). The input in the `MLML` function are matrices with methylated and unmethylated information from each conversion method. We can use the *MethylSet* class, which contains the methylated and unmethylated signals. The most basic way to construct a *MethylSet* is using the function `preprocessRaw`.

Here we chose the function `preprocessNoob` (Triche et al. 2013) for background correction, dye bias normalization and construction of the *MethylSet*. We encourage the user to consider other normalization methods such as SWAN (Maksimovic, Gordon, and Oshlack 2012), BMIQ (Teschendorff et al. 2012), RCP (Niu, Xu, and Taylor 2016), Funnorm (Fortin et al. 2014), and others, as well as combination of some of these methods, as suggested by J. Liu and Siegmund (2016).

The BS replicates are in columns 1, 3, 5, and 6 (information from `pD$title`). The remaining columns are from the oxBS treated replicates.

```
BSindex <- c(1,3,5,6)
oxBSindex <- c(7,8,2,4)

MSet.noob <- preprocessNoob(rgSet=rgSet)
```

After the preprocessing steps we can use `MLML` from the *MLML2R* package.

```
MChannelBS <- getMeth(MSet.noob)[,BSindex]
UChannelBS <- getUnmeth(MSet.noob)[,BSindex]
MChannelOxBS <- getMeth(MSet.noob)[,oxBSindex]
UChannelOxBS <- getUnmeth(MSet.noob)[,oxBSindex]
```

When only two methods are available, the default option of `MLML` function returns the exact constrained maximum likelihood estimates using the the pool-adjacent-violators algorithm (PAVA) (Ayer et al. 1955).

```
results_exact <- MLML(T.matrix = MChannelBS , U.matrix = UChannelBS,
                      L.matrix = UChannelOxBS, M.matrix = MChannelOxBS)

save(results_exact,file="results_exact_oxBS.rds")
```

Maximum likelihood estimate via EM-algorithm approach (Qu et al. 2013) is obtained with the option `iterative=TRUE`. In this case, the default (or user specified) `tol` is considered in the iterative method.

```
results_em <- MLML(T.matrix = MChannelBS , U.matrix = UChannelBS,
                   L.matrix = UChannelOxBS, M.matrix = MChannelOxBS,
                   iterative = TRUE)
```

The estimates are very similar for both methods:

```
all.equal(results_exact$hmC,results_em$hmC,scale=1)
```

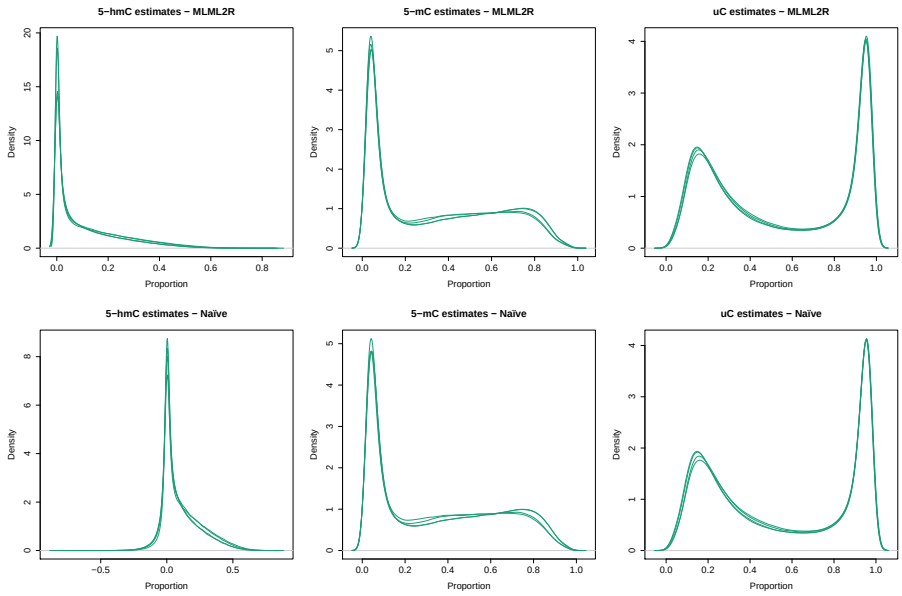

Figure 1: **Estimated proportions of 5-hmC, 5-mC and uC for the CpGs in the dataset from Field (2015) using the MLML function with default (PAVA) options (top row) and the naïve (subtraction) method (bottom row)**

2.2 Publicly available array data: TAB and BS methods

We will use the dataset from Thienpont et al. (2016), which consists of 24 DNA samples treated with TAB-BS and hybridized to the Infinium 450K array from newly diagnosed and untreated non-small-cell lung cancer patients (12 normoxic and 12 hypoxic tumours). The dataset is deposited under GEO accession number GSE71398.

We will need the following packages:

```
library(MLML2R)
library(minfi)
library(GEOquery)
library(IlluminaHumanMethylation450kmanifest)
```

Obtaining the data:

```
getGEOSuppFiles("GSE71398")
untar("GSE71398/GSE71398_RAW.tar", exdir = "GSE71398/idat")

list.files("GSE71398/idat", pattern = "idat")
files <- list.files("GSE71398/idat", pattern = "idat.gz$", full = TRUE)
sapply(files, gunzip, overwrite = TRUE)
```

Reading the `.IDAT` files:

```
rgSet <- read.metharray.exp("GSE71398/idat")
```

The phenotype data is not yet available from the `rgSet`.

```
pData(rgSet)
```

We need to correctly identify the 24 DNA samples: 12 normoxic and 12 hypoxic non-small-cell lung cancer. We also need the information about the conversion method used in each replicate: BS or TAB. We will access this information automatically from GEO:

```
if (!file.exists("GSE71398/GSE71398_series_matrix.txt.gz"))
download.file(
"https://ftp.ncbi.nlm.nih.gov/geo/series/GSE71nnn/GSE71398/matrix/GSE71398_series_matrix.txt.gz",
"GSE71398/GSE71398_series_matrix.txt.gz")

geoMat <- getGEO(filename="GSE71398/GSE71398_series_matrix.txt.gz",getGPL=FALSE)
pD.all <- pData(geoMat)

#Another option
#geoMat <- getGEO("GSE71398")
#pD.all <- pData(geoMat[[1]])

pD <- pD.all[, c("title", "geo_accession", "source_name_ch1",
                 "tabchip or bschip:ch1","hypoxia status:ch1",
                 "tumor name:ch1","batch:ch1","platform_id")]
pD$method <- pD$`tabchip or bschip:ch1`
pD$group <- pD$`hypoxia status:ch1`
pD$sample <- pD$`tumor name:ch1`
pD$batch <- pD$`batch:ch1`
```

This phenotype data needs to be merged into the methylation data. The following commands guarantee we have the same replicate identifier in both datasets before merging.

```
sampleNames(rgSet) <- sapply(sampleNames(rgSet),function(x)
  strsplit(x,"_")[[1]][1])
rownames(pD) <- as.character(pD$geo_accession)
```

```
pD <- pD[sampleNames(rgSet),]
pData(rgSet) <- as(pD,"DataFrame")
rgSet
```

The following command produces a quality control report, which helps to identify failed samples:

```
qcReport(rgSet, pdf= "qcReport_tab_bs.pdf")
```

After looking at the quality control report, we notice a problematic sample: GSM1833667. This sample and its corresponding pair in the TAB experiment, GSM1833691, were removed from subsequent analysis.

The input in the `MLML` function accepts as input a *MethylSet*, which contains the methylated and unmethylated signals. We used the function `preprocessNoob` (Triche et al. 2013) for background correction, dye bias normalization and construction of the *MethylSet*.

```
BSindex <- which(pD$method=="BSchip")[-which(pD$geo_accession
                                             %in% c("GSM1833667","GSM1833691"))]
TABindex <- which(pD$method=="TABchip")[-which(pD$geo_accession
                                               %in% c("GSM1833667","GSM1833691"))]

MSet.noob <- preprocessNoob(rgSet)

MChannelBS <- getMeth(MSet.noob)[,BSindex]
UChannelBS <- getUnmeth(MSet.noob)[,BSindex]
MChannelTAB <- getMeth(MSet.noob)[,TABindex]
UChannelTAB <- getUnmeth(MSet.noob)[,TABindex]
```

We can now use `MLML` from the *MLML2R* package.

One needs to carefully check if the columns across the different input matrices represent the same sample. In this example, all matrices have the samples consistently represented in the columns: sample 1 in the first column, sample 2 in the second, and so forth.

When any two of the methods are available, the default option of `MLML` function returns the exact constrained maximum likelihood estimates using the the pool-adjacent-violators algorithm (PAVA) (Ayer et al. 1955).

```
results_exact <- MLML(T.matrix = MChannelBS , U.matrix = UChannelBS,
                      G.matrix = UChannelTAB, H.matrix = MChannelTAB)
```

Maximum likelihood estimate via EM-algorithm approach (Qu et al. 2013) is obtained with the option `iterative=TRUE`. In this case, the default (or user specified) `tol` is considered in the iterative method.

```
results_em <- MLML(T.matrix = MChannelBS , U.matrix = UChannelBS,
                   G.matrix = UChannelTAB, H.matrix = MChannelTAB,
                   iterative = TRUE)
```

The estimates for 5-hmC proportions are very similar for both methods:

```
all.equal(results_exact$hmC,results_em$hmC,scale=1)
```

The estimates for 5-mC proportions are very similar for both methods:

```
all.equal(results_exact$mC,results_em$mC,scale=1)
```

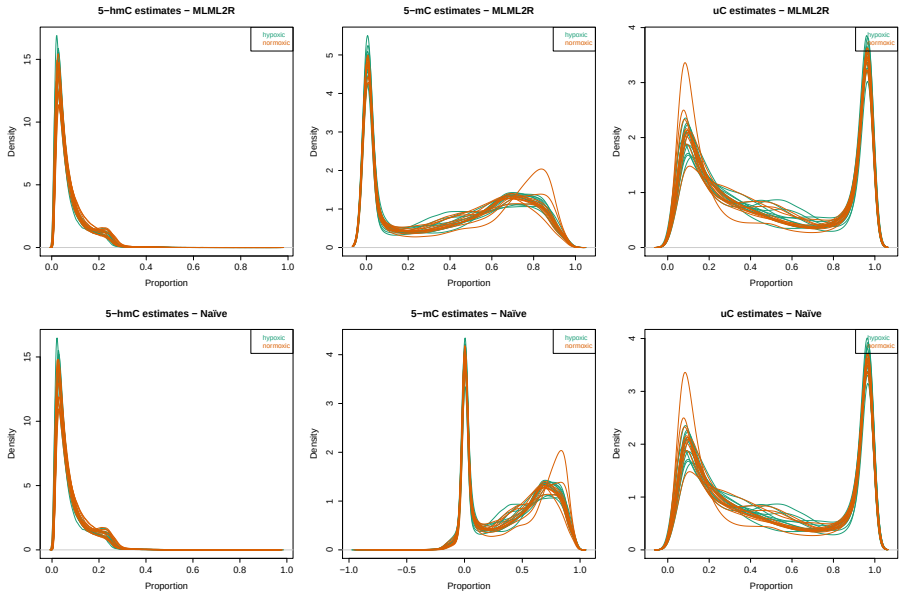

Figure 2: **Estimated proportions of 5-hmC, 5-mC and uC for the CpGs in the dataset from Thienpont et al (2016), using the MLML function with default (PAVA) options (top row) and the naïve (subtraction) method (bottom row)**

2.3 Publicly available sequencing data: oxBS and BS methods

We will use the dataset from Li et al. (2016), which consists of three human lung normal-tumor pairs (six samples). Each sample was divided into two replicates: one treated with BS and the other with oxBS, which were then sequenced using the Illumina HiSeq 2000 (Homo sapiens) platform. The preprocessed dataset is available at GEO accession GSE70090. The details of the preprocessing procedures are described in Li et al. (2016).

Obtaining the data:

```
library(GEOquery)

getGEOSuppFiles("GSE70090")
untar("GSE70090/GSE70090_RAW.tar", exdir = "GSE70090/data")
```

Decompressing the files:

```
dataFiles <- list.files("GSE70090/data", pattern = "txt.gz$", full = TRUE)
sapply(dataFiles, gunzip, overwrite = TRUE)
```

We need to identify the different samples from different methods (BS-conversion, oxBS-conversion), we will use the file names do extract this information.

```
files <- list.files("GSE70090/data")
filesfull <- list.files("GSE70090/data",full=TRUE)
tissue <- sapply(files,function(x) strsplit(x,"_")[[1]][2]) # tissue
id <- sapply(files,function(x) strsplit(x,"_")[[1]][3]) # sample id
tmp <- sapply(files,function(x) strsplit(x,"_")[[1]][4])
convMeth <- sapply(tmp, function(x) strsplit(x,"\\.")[[1]][1]) # DNA conversion method
group <- ifelse(id %in% c("N1","N2","N3","N4"),"normal","tumor")
id2 <- paste(tissue,id,sep="_")
GSM <- sapply(files,function(x) strsplit(x,"_")[[1]][1]) # GSM
pheno <- data.frame(GSM=GSM,tissue=tissue,id=id2,convMeth=convMeth,
                    group=group,file=filesfull,stringsAsFactors = FALSE)
```

Selecting only the three human lung normal-tumor pairs:

```
library(data.table)

phenoLung <- pheno[pheno$tissue=="lung",]

# order to have all BS samples and then all oxBS samples
phenoLung <- phenoLung[order(phenoLung$convMeth,phenoLung$id),]
```

Preparing the data for input in the `MLML` function:

```
### BS
files <- phenoLung$file[phenoLung$convMeth=="BS"]
C_BS    <- do.call(cbind,lapply(files,function(fn)
  fread(fn,data.table=FALSE,select=c("methylated_read_count"))))
TotalBS    <- do.call(cbind,lapply(files,function(fn)
  fread(fn,data.table=FALSE,select=c("total_read_count"))))
T_BS <- TotalBS - C_BS


### oxBS
files <- phenoLung$file[phenoLung$convMeth=="oxBS"]
C_OxBS    <- do.call(cbind,lapply(files,function(fn)
  fread(fn,data.table=FALSE,select=c("methylated_read_count"))))
TotalOxBS    <- do.call(cbind,lapply(files,function(fn)
  fread(fn,data.table=FALSE,select=c("total_read_count"))))
T_OxBS <- TotalOxBS - C_OxBS

# since rownames and colnames are the same across files:
tmp <- fread(files[1], data.table=FALSE, select=c("chr","position"))
CpG <- paste(tmp[,1],tmp[,2],sep="-")

rownames(C_BS) <- CpG
rownames(T_BS) <- CpG

colnames(C_BS) <- phenoLung$id[phenoLung$convMeth=="BS"]
colnames(T_BS) <- phenoLung$id[phenoLung$convMeth=="BS"]

rownames(C_OxBS) <- CpG
```

```
rownames(T_OxBS) <- CpG

colnames(C_OxBS) <- phenoLung$id[phenoLung$convMeth=="oxBS"]
colnames(T_OxBS) <- phenoLung$id[phenoLung$convMeth=="oxBS"]

Tm = as.matrix(C_BS)
Um = as.matrix(T_BS)
Lm = as.matrix(T_OxBS)
Mm = as.matrix(C_OxBS)
```

Only CpGs with coverage of at least 10 across all samples and all conversion procedures (BS and oxBS) were considered in the following results (7685557 CpGs).

```
TotalBS <- Tm+Um
TotalOxBS <- Lm+Mm

library(matrixStats)

tmp1 <- rowMins(TotalBS,na.rm=TRUE) # minimum coverage across samples from BS for each CpG
tmp2 <- rowMins(TotalOxBS,na.rm=TRUE) # minimum coverage across samples from oxBS for each CpG

aa <-which(tmp1>=10 & tmp2>=10)
# CpGs with coverage at least 10 across all samples for both methods (BS and oxBS)
length(aa)
```

We can now use `MLML` from the *MLML2R* package.

```
library(MLML2R)

results_exact <- MLML(T.matrix = Tm[aa,],
        U.matrix = Um[aa,],
        L.matrix = Lm[aa,],
        M.matrix = Mm[aa,])

results_em <- MLML(T.matrix = Tm[aa,],
        U.matrix = Um[aa,],
        L.matrix = Lm[aa,],
        M.matrix = Mm[aa,],
        iterative = TRUE)
```

Comparing the estimates for 5-hmC proportions from iterative and non iterative option from `MLML` function:

```
all.equal(results_exact$hmC,results_em$hmC,scale=1)
```

The estimates for 5-mC proportions are also very similar for both methods:

```
all.equal(results_exact$mC,results_em$mC,scale=1)
```

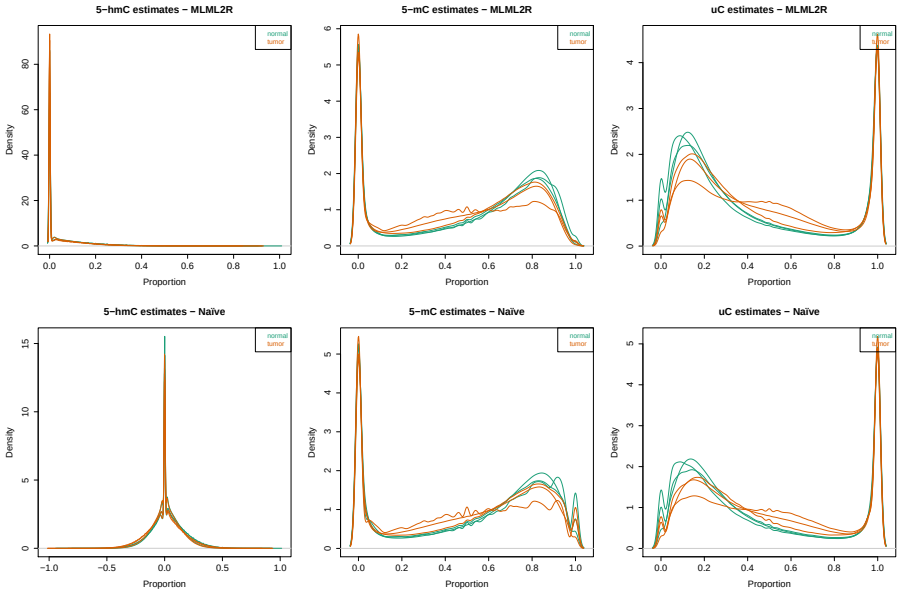

Figure 3: **Estimated proportions of 5-hmC, 5-mC and uC for the CpGs in the dataset from Li et al (2016) using the MLML function with default options (top row) and the naïve method (bottom row)**

2.4 Simulated data

To illustrate the package when all the three methods are available or when any combination of only two of them are available, we will simulate a dataset.

We will use a sample of the estimates of 5-mC, 5-hmC and uC of the previous oxBS+BS array example shown in Section 2.1 as the true proportions, as shown in Figure 4.

Two replicate samples with 1000 CpGs will be simulated. For CpG i in sample j:

$$
\begin{aligned}
T_{i,j} &\sim Binomial(n = c_{i,j}, p = p_m + p_h) \\
M_{i,j} &\sim Binomial(n = c_{i,j}, p = p_m) \\
H_{i,j} &\sim Binomial(n = c_{i,j}, p = p_h) \\
U_{i,j} &= c_{i,j} - T_{i,j} \\
L_{i,j} &= c_{i,j} - M_{i,j} \\
G_{i,j} &= c_{i,j} - H_{i,j}
\end{aligned}
$$

where the random variables are defined in Table 1, and $c_{i,j}$ represents the coverage for CpG i in sample j.

The following code produce the simulated data:

```
load("results_exact_oxBS.rds") # load estimates from previous example

set.seed(112017)

index <- sample(1:dim(results_exact$mC)[1],1000,replace=FALSE) # 1000 CpGs
```

```
Coverage <- round(MChannelBS+UChannelBS)[index,1:2] # considering 2 samples

temp1 <- data.frame(n=as.vector(Coverage),
                    p_m=c(results_exact$mC[index,1],
                          results_exact$mC[index,1]),
                    p_h=c(results_exact$hmC[index,1],
                          results_exact$hmC[index,1]))

MChannelBS_temp <- c()
for (i in 1:dim(temp1)[1])
{
  MChannelBS_temp[i] <- rbinom(n=1, size=temp1$n[i],
                               prob=(temp1$p_m[i]+temp1$p_h[i]))
}


UChannelBS_sim2 <- matrix(Coverage - MChannelBS_temp,ncol=2)
MChannelBS_sim2 <- matrix(MChannelBS_temp,ncol=2)


MChannelOxBS_temp <- c()
for (i in 1:dim(temp1)[1])
{
  MChannelOxBS_temp[i] <- rbinom(n=1, size=temp1$n[i], prob=temp1$p_m[i])
}

UChannelOxBS_sim2 <- matrix(Coverage - MChannelOxBS_temp,ncol=2)
MChannelOxBS_sim2 <- matrix(MChannelOxBS_temp,ncol=2)


MChannelTAB_temp <- c()
for (i in 1:dim(temp1)[1])
{
  MChannelTAB_temp[i] <- rbinom(n=1, size=temp1$n[i], prob=temp1$p_h[i])
}


UChannelTAB_sim2 <- matrix(Coverage - MChannelTAB_temp,ncol=2)
MChannelTAB_sim2 <- matrix(MChannelTAB_temp,ncol=2)

true_parameters_sim2 <- data.frame(p_m=results_exact$mC[index,1],
                                   p_h=results_exact$hmC[index,1])
true_parameters_sim2$p_u <- 1-true_parameters_sim2$p_m-true_parameters_sim2$p_h
```

2.4.1 BS and oxBS methods

When only two methods are available, the default option returns the exact constrained maximum likelihood estimates using the the pool-adjacent-violators algorithm (PAVA) (Ayer et al. 1955).

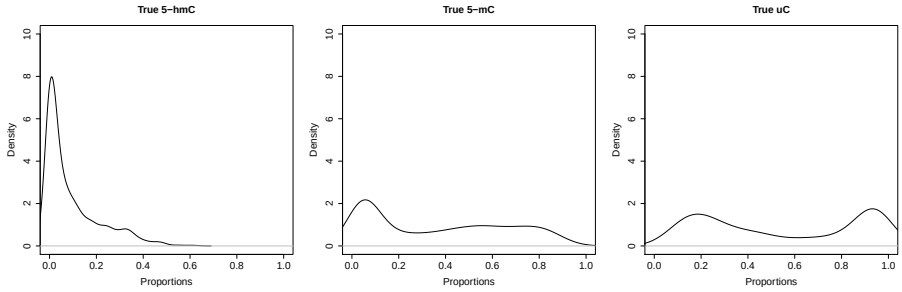

Figure 4: True proportions of hydroxymethylation, methylation and unmethylation for the CpGs used to generate the datasets

```
library(MLML2R)
 results_exactBO1 <- MLML(T.matrix = MChannelBS_sim2,
                          U.matrix = UChannelBS_sim2,
                          L.matrix = UChannelOxBS_sim2,
                          M.matrix = MChannelOxBS_sim2)
```

Maximum likelihood estimate via EM-algorithm approach (Qu et al. 2013) is obtained with the option `iterative=TRUE`. In this case, the default (or user specified) `tol` is considered in the iterative method.

```
 results_emBO1 <- MLML(T.matrix = MChannelBS_sim2,
                       U.matrix = UChannelBS_sim2,
                       L.matrix = UChannelOxBS_sim2,
                       M.matrix = MChannelOxBS_sim2,
                       iterative=TRUE)
```

When only two methods are available, we highly recommend the default option `iterative=FALSE` since the difference in the estimates obtained via EM and exact constrained is very small, but the former requires more computational effort:

```
 all.equal(results_emBO1$hmC,results_exactBO1$hmC,scale=1)
## [1] "Mean absolute difference: 0.0001259144"
```

```
 library(microbenchmark)
 mbmBO1 = microbenchmark(
    EXACT = MLML(T.matrix = MChannelBS_sim2,
                 U.matrix = UChannelBS_sim2,
                 L.matrix = UChannelOxBS_sim2,
                 M.matrix = MChannelOxBS_sim2),
    EM =    MLML(T.matrix = MChannelBS_sim2,
                 U.matrix = UChannelBS_sim2,
                 L.matrix = UChannelOxBS_sim2,
                 M.matrix = MChannelOxBS_sim2,
                 iterative=TRUE),
    times=10)
 mbmBO1
## Unit: microseconds
##   expr       min        lq       mean    median        uq       max neval
```

```
##  EXACT   405.638   413.482   550.3115   505.848   709.989   775.382    10
##     EM 11557.870 13317.750 18084.4434 15925.426 22909.628 27660.043    10
```

Comparison between approximate exact constrained and true hydroxymethylation proportion used in simulation:

```
all.equal(true_parameters_sim2$p_h,results_exactBO1$hmC[,1],scale=1)
## [1] "Mean absolute difference: 0.005980957"
```

Comparison between EM-algorithm and true hydroxymethylation proportion used in simulation:

```
all.equal(true_parameters_sim2$p_h,results_emBO1$hmC[,1],scale=1)
## [1] "Mean absolute difference: 0.005396121"
```

2.4.2 BS and TAB methods

Using PAVA:

```
results_exactBT1 <- MLML(T.matrix = MChannelBS_sim2,
                         U.matrix = UChannelBS_sim2,
                         G.matrix = UChannelTAB_sim2,
                         H.matrix = MChannelTAB_sim2)
```

Using EM-algorithm:

```
 results_emBT1 <- MLML(T.matrix = MChannelBS_sim2,
                       U.matrix = UChannelBS_sim2,
                       G.matrix = UChannelTAB_sim2,
                       H.matrix = MChannelTAB_sim2,
                       iterative=TRUE)
```

Comparison between PAVA and EM:

```
 all.equal(results_emBT1$hmC,results_exactBT1$hmC,scale=1)
## [1] "Mean absolute difference: 3.196297e-07"
```

```
 mbmBT1 = microbenchmark(
    EXACT = MLML(T.matrix = MChannelBS_sim2,
                 U.matrix = UChannelBS_sim2,
                 G.matrix = UChannelTAB_sim2,
                 H.matrix = MChannelTAB_sim2),
    EM =    MLML(T.matrix = MChannelBS_sim2,
                 U.matrix = UChannelBS_sim2,
                 G.matrix = UChannelTAB_sim2,
                 H.matrix = MChannelTAB_sim2,
                 iterative=TRUE),
    times=10)
 mbmBT1
## Unit: microseconds
##   expr      min       lq       mean   median        uq       max neval
##  EXACT  291.817  318.118   339.8806  350.620   368.409   368.549    10
```

```
##     EM 7820.834 7870.722 10499.6755 8155.428 14333.196 17215.251    10
```

Comparison between approximate exact constrained and true hydroxymethylation proportion used in simulation:

```
all.equal(true_parameters_sim2$p_h,results_exactBT1$hmC[,1],scale=1)
## [1] "Mean absolute difference: 0.0030728"
```

Comparison between EM-algorithm and true hydroxymethylation proportion used in simulation:

```
all.equal(true_parameters_sim2$p_h,results_emBT1$hmC[,1],scale=1)
## [1] "Mean absolute difference: 0.002319746"
```

2.4.3 oxBS and TAB methods

Using PAVA:

```
 results_exactOT1 <- MLML(L.matrix = UChannelOxBS_sim2,
                          M.matrix = MChannelOxBS_sim2,
                          G.matrix = UChannelTAB_sim2,
                          H.matrix = MChannelTAB_sim2)
```

Using EM-algorithm:

```
 results_emOT1 <- MLML(L.matrix = UChannelOxBS_sim2,
                       M.matrix = MChannelOxBS_sim2,
                       G.matrix = UChannelTAB_sim2,
                       H.matrix = MChannelTAB_sim2,
                       iterative=TRUE)
```

Comparison between PAVA and EM:

```
 all.equal(results_emOT1$hmC,results_exactOT1$hmC,scale=1)
## [1] "Mean absolute difference: 1.435988e-07"
```

```
 mbmOT1 = microbenchmark(
    EXACT = MLML(L.matrix = UChannelOxBS_sim2,
                 M.matrix = MChannelOxBS_sim2,
                 G.matrix = UChannelTAB_sim2,
                 H.matrix = MChannelTAB_sim2),
    EM =    MLML(L.matrix = UChannelOxBS_sim2,
                 M.matrix = MChannelOxBS_sim2,
                 G.matrix = UChannelTAB_sim2,
                 H.matrix = MChannelTAB_sim2,
                 iterative=TRUE),
    times=10)
 mbmOT1
## Unit: microseconds
##   expr      min       lq      mean   median       uq       max neval
##  EXACT  287.558  351.949  405.8785  359.127  385.559   640.625    10
##     EM 4067.731 4119.211 5316.3666 4213.685 4650.530 12967.650    10
```

Comparison between approximate exact constrained and true 5-hmC proportion used in simulation:

```
all.equal(true_parameters_sim2$p_h,results_exactOT1$hmC[,1],scale=1)
## [1] "Mean absolute difference: 0.0030728"
```

Comparison between EM-algorithm and true 5-hmC proportion used in simulation:

```
all.equal(true_parameters_sim2$p_h,results_emOT1$hmC[,1],scale=1)
## [1] "Mean absolute difference: 0.003072645"
```

2.4.4 BS, oxBS and TAB methods

When data from the three methods are available, the default otion in the `MLML` function returns the constrained maximum likelihood estimates using an approximated solution for Lagrange multipliers method.

```
results_exactBOT1 <- MLML(T.matrix = MChannelBS_sim2,
                          U.matrix = UChannelBS_sim2,
                          L.matrix = UChannelOxBS_sim2,
                          M.matrix = MChannelOxBS_sim2,
                          G.matrix = UChannelTAB_sim2,
                          H.matrix = MChannelTAB_sim2)
```

Maximum likelihood estimate via EM-algorithm approach (Qu et al. 2013) is obtained with the option `iterative=TRUE`. In this case, the default (or user specified) `tol` is considered in the iterative method.

```
 results_emBOT1 <- MLML(T.matrix = MChannelBS_sim2,
                        U.matrix = UChannelBS_sim2,
                        L.matrix = UChannelOxBS_sim2,
                        M.matrix = MChannelOxBS_sim2,
                        G.matrix = UChannelTAB_sim2,
                        H.matrix = MChannelTAB_sim2,iterative=TRUE)
```

We recommend the default option `iterative=FALSE` since the difference in the estimates obtained via EM and the approximate exact constrained is very small, but the former requires more computational effort:

```
 all.equal(results_emBOT1$hmC,results_exactBOT1$hmC,scale=1)
## [1] "Mean absolute difference: 6.665856e-07"
```

```
 mbmBOT1 = microbenchmark(
    EXACT = MLML(T.matrix = MChannelBS_sim2,
                 U.matrix = UChannelBS_sim2,
                 L.matrix = UChannelOxBS_sim2,
                 M.matrix = MChannelOxBS_sim2,
                 G.matrix = UChannelTAB_sim2,
                 H.matrix = MChannelTAB_sim2),
    EM =    MLML(T.matrix = MChannelBS_sim2,
                 U.matrix = UChannelBS_sim2,
```

```
                  L.matrix = UChannelOxBS_sim2,
                  M.matrix = MChannelOxBS_sim2,
                  G.matrix = UChannelTAB_sim2,
                  H.matrix = MChannelTAB_sim2,
                  iterative=TRUE),
    times=10)
 mbmBOT1
## Unit: microseconds
##   expr      min       lq     mean   median       uq       max neval
##  EXACT  866.178  978.404 1188.448 1011.323 1492.131  1730.348    10
##     EM 1867.591 2139.370 3087.105 2207.530 2413.707 11115.167    10
```

Comparison between approximate exact constrained and true hydroxymethylation proportion used in simulation:

```
all.equal(true_parameters_sim2$p_h,results_exactBOT1$hmC[,1],scale=1)
## [1] "Mean absolute difference: 0.002708598"
```

Comparison between EM-algorithm and true hydroxymethylation proportion used in simulation:

```
all.equal(true_parameters_sim2$p_h,results_emBOT1$hmC[,1],scale=1)
## [1] "Mean absolute difference: 0.002045009"
```

References

Aryee, Martin J., Andrew E. Jaffe, Hector Corrada-Bravo, Christine Ladd-Acosta, Andrew P. Feinberg, Kasper D. Hansen, and Rafael A. Irizarry. 2014. "Minfi: A flexible and comprehensive Bioconductor package for the analysis of Infinium DNA Methylation microarrays." *Bioinformatics* 30 (10): 1363–9. doi:10.1093/bioinformatics/btu049.

Ayer, Miriam, H. D. Brunk, G. M. Ewing, W. T. Reid, and Edward Silverman. 1955. "An Empirical Distribution Function for Sampling with Incomplete Information." *Ann. Math. Statist.* 26 (4). The Institute of Mathematical Statistics: 641–47. doi:10.1214/aoms/1177728423.

Field, Dario AND Bachman, Sarah F. AND Beraldi. 2015. "Accurate Measurement of 5-Methylcytosine and 5-Hydroxymethylcytosine in Human Cerebellum Dna by Oxidative Bisulfite on an Array (Oxbs-Array)." *PLOS ONE* 10 (2). Public Library of Science: 1–12. doi:10.1371/journal.pone.0118202.

Fortin, Jean-Philippe, Aurelie Labbe, Mathieu Lemire, Brent W. Zanke, Thomas J. Hudson, Elana J. Fertig, Celia M.T. Greenwood, and Kasper D. Hansen. 2014. "Functional Normalization of 450k Methylation Array Data Improves Replication in Large Cancer Studies." *Genome Biology* 15 (12): 503. doi:10.1186/s13059-014-0503-2.

Li, Xin, Yun Liu, Tal Salz, Kasper D. Hansen, and Andrew Feinberg. 2016. "Whole-Genome Analysis of the Methylome and Hydroxymethylome in Normal and Malignant Lung and Liver." *Genome Research* 26 (12). Cold Spring Harbor Laboratory: 1730–41. doi:10.1101/gr.211854.116.

Liu, Jie, and Kimberly D. Siegmund. 2016. "An Evaluation of Processing Methods for HumanMethylation450 BeadChip Data." *BMC Genomics* 17 (1). Springer Nature. doi:10.1186/s12864-016-2819-7.

Maksimovic, Jovana, Lavinia Gordon, and Alicia Oshlack. 2012. "SWAN: Subset-Quantile Within Array Normalization for Illumina Infinium HumanMethylation450 BeadChips." *Genome Biology* 13 (6). Springer Nature: R44. doi:10.1186/gb-2012-13-6-r44.

Niu, Liang, Zongli Xu, and Jack A. Taylor. 2016. "RCP: A Novel Probe Design Bias Correction Method for Illumina Methylation BeadChip." *Bioinformatics* 32 (17). Oxford University Press (OUP): 2659–63. doi:10.1093/bioinformatics/btw285.

Qu, Jianghan, Meng Zhou, Qiang Song, Elizabeth E. Hong, and Andrew D. Smith. 2013. "MLML: Consistent Simultaneous Estimates of Dna Methylation and Hydroxymethylation." *Bioinformatics* 29 (20): 2645–6. doi:10.1093/bioinformatics/btt459.

Teschendorff, Andrew E., Francesco Marabita, Matthias Lechner, Thomas Bartlett, Jesper Tegner, David Gomez-Cabrero, and Stephan Beck. 2012. "A Beta-Mixture Quantile Normalization Method for Correcting Probe Design Bias in Illumina Infinium 450 K DNA Methylation Data." *Bioinformatics* 29 (2). Oxford University Press (OUP): 189–96. doi:10.1093/bioinformatics/bts680.

Thienpont, Bernard, Jessica Steinbacher, Hui Zhao, Flora D'Anna, Anna Kuchnio, Athanasios Ploumakis, Bart Ghesquière, et al. 2016. "Tumour Hypoxia Causes DNA Hypermethylation by Reducing TET Activity." *Nature* 537 (7618). Springer Nature: 63–68. doi:10.1038/nature19081.

Triche, Timothy J., Daniel J. Weisenberger, David Van Den Berg, Peter W. Laird, and Kimberly D. Siegmund. 2013. "Low-Level Processing of Illumina Infinium DNA Methylation BeadArrays." *Nucleic Acids Research* 41 (7): e90. doi:10.1093/nar/gkt090.