methylSig: A package for whole genome DNA methylation analysis

Yongseok Park, Maria E. Figueroa, Laura S. Rozek, and Maureen A. Sartor

2015-10-08

Contents

1 Introduction

DNA methylation plays critical roles in gene regulation and cellular specification without altering DNA sequences. It is one of the best understood and most intensively studied epigenetic marks in mammalian cells. Treatment of DNA with sodium bisulfite deaminates unmethylated cytosines to uracil while methylated cytosines are resistant to this conversion thus allowing for the discrimination between methylated and unmethylated CpG sites. Sodium bisulfite pre-treatment of DNA coupled with next-generation sequencing has allowed DNA methylation to be studied quantitatively and genome-wide at single cytosine site resolution.

`methylSig` is a method for testing for differential methylated cytosines (DMCs) or regions (DMRs) in whole-genome bisulfite sequencing (bis-seq) or reduced representation bisulfite sequencing (RRBS) experiments. `methylSig` uses a beta binomial model to test for significant differences between groups of samples. Several options exist for either site-specific or sliding window tests, combining strands, and for variance estimation. It allows annotating the resulting regions to multiple genome features, and visualizing the results for chosen genomic regions.

This document provides a step by step guide for the `methylSig` package.

2 Installation

`methylSig` is available on GitHub at http://www.github.com/sartorlab/methylSig, and the easiest way to install it is as follows:

```
library(devtools)
install_github('sartorlab/methylSig')
```

3 Basic usage

3.1 Methylation score files

`methylSig` expects input data to be formatted as follows:

```
##         chrBase   chr     base strand coverage  freqC freqT
##  chr21.43008527 chr21 43008527      F       32 100.00  0.00
##  chr21.43008531 chr21 43008531      F       32  96.88  3.12
##  chr21.43008543 chr21 43008543      F       32  90.62  9.38
##  chr21.43008674 chr21 43008674      R       27 100.00  0.00
##  chr21.43008710 chr21 43008710      R       67  94.03  5.97
##  chr21.43008720 chr21 43008720      R       67  92.54  7.46
```

Such CpG methylation score files can be obtained using `bismark` with the `--bedGraph --cytosine_report` flags in the `bismark_methylation_extractor`. The following `awk` command can be used on each cytosine report to obtain the correct format. Note, in `$4 + $5 > 10`, the 10 refers to the desired minimum coverage at each CpG site.

```
awk -v OFS="\t" '$4 + $5 > 10 {print $1"."$2, $1, $2, $3, $4 + $5, $4, $5}' in > out
```

The CpG methylation score file must contain at least seven columns. Among these, second to seventh column must be, in order, chromosome, base, strand, coverage, percentage of Cs and percentage of Ts. Column names are not important. Strand format is F/R or +/-, where F/+ represents forward and R/- represents reverse strand.

3.2 Reading methylation score files

`methylSig` package provides the `methylSigReadData()`function to read CpG methylation score files and convert these files into a `methylSigData` object for further analysis and annotation. The parameters and default options are:

```
methylSigReadData(fileList, sample.ids, assembly = NA, pipeline = NA,
  header = TRUE, context = NA, resolution = "base", treatment,
  destranded = TRUE, maxCount = 500, minCount = 10, filterSNPs = FALSE,
  num.cores = 1, quiet = FALSE)
}
```

Using data built into `methylSig` a typical read call might look like:

```
fileList = c(system.file("extdata", "AML_1.txt", package = "methylSig"),
             system.file("extdata", "AML_2.txt", package = "methylSig"),
             system.file("extdata", "AML_3.txt", package = "methylSig"),
             system.file("extdata", "AML_4.txt", package = "methylSig"),
             system.file("extdata", "NBM_1.txt", package = "methylSig"),
             system.file("extdata", "NBM_2.txt", package = "methylSig"),
             system.file("extdata", "NBM_3.txt", package = "methylSig"),
             system.file("extdata", "NBM_4.txt", package = "methylSig"))


sample.id = c("AML1", "AML2", "AML3", "AML4", "NBM1", "NBM2", "NBM3", "NBM4")

treatment = c(1,1,1,1,0,0,0,0)
#### Read Data ####
meth <- methylSigReadData(fileList, sample.ids = sample.id, assembly = "hg18",
           treatment = treatment, context = "CpG", destranded=TRUE)
```

```
## Reading file (1/8) -- /Users/raymond/Library/R/3.2/library/methylSig/extdata/AML_1.txt
## (1/8) Count Invalid List: 0/2411=0
## Reading file (2/8) -- /Users/raymond/Library/R/3.2/library/methylSig/extdata/AML_2.txt
## (2/8) Count Invalid List: 11/4259=0.00258
## Reading file (3/8) -- /Users/raymond/Library/R/3.2/library/methylSig/extdata/AML_3.txt
## (3/8) Count Invalid List: 39/3861=0.0101
## Reading file (4/8) -- /Users/raymond/Library/R/3.2/library/methylSig/extdata/AML_4.txt
## (4/8) Count Invalid List: 2/3750=0.000533
## Reading file (5/8) -- /Users/raymond/Library/R/3.2/library/methylSig/extdata/NBM_1.txt
## (5/8) Count Invalid List: 0/6228=0
## Reading file (6/8) -- /Users/raymond/Library/R/3.2/library/methylSig/extdata/NBM_2.txt
## (6/8) Count Invalid List: 0/6430=0
## Reading file (7/8) -- /Users/raymond/Library/R/3.2/library/methylSig/extdata/NBM_3.txt
## (7/8) Count Invalid List: 1/5962=0.000168
## Reading file (8/8) -- /Users/raymond/Library/R/3.2/library/methylSig/extdata/NBM_4.txt
## (8/8) Count Invalid List: 23/3694=0.00623
## (1)(2)(3)(4)(5)(6)(7)(8)
```

It is possible for the user to filter out CpG sites based on the read coverage. CpG sites with very large read coverage may be due to PCR bias and hence including CpG sites with very high coverage may distort the statistics of data analysis. The `methylSigReadData()` function provides `minCount` and `maxCount` arguments for defining lower and upper limits for coverage. The default values are 10 and 500 respectively. It is also

possible to exclude C > T SNPs determined by the 1000 Genomes Project with the `filterSNPs` option. This is not done by default.

There are many arguments for the `methylSigReadData()` funciton. Among these `fileList`, `sample.ids` and `treatment` are required. Some options have default values, for example, `destranded=TRUE`, `num.cores=1`, and `quiet=FALSE`. Other arguments such as `assembly`, `context` and `pipeline` are optional and for information purposes only. The data type of `treatment` is a numeric vector. Each number represents a group. Multiple groups can be stored in one `methylSigData` object.

The arguemnt `num.cores` is used for multi-thread reading.

3.3 Differential methylation analysis

The main function of this package is the differential methylation analysis function `methySigCalc()`. It calculates differential methylation statistics between two groups of samples. It uses a beta-binomial approach to calculate differential methylation statistics, accounting for coverage and variation among samples within each group.

```
methylSigCalc(meth, groups = c(Treatment = 1, Control = 0),
  dispersion = "both", local.disp = FALSE, winsize.disp = 200,
  local.meth = FALSE, winsize.meth = 200, min.per.group = c(3, 3),
  weightFunc = methylSig_weightFunc, T.approx = TRUE, num.cores = 1)
}
```

3.4 Site specific analysis

The default is to do site specific analysis and to use both groups to estimate variances.

```
myDiffSigboth = methylSigCalc(meth, groups=c(1,0), min.per.group=3)
```

```
## Total number of bases: 3.26k
```

The differentially methylated cytosines (DMCs) can be defined based on qvalues, pvalues and the methylation rate difference between two tested groups.

```
myDiffSigbothDMCs = myDiffSigboth[myDiffSigboth[,"qvalue"] <= 0.05
                                  & abs(myDiffSigboth[,"meth.diff"])>=25, ]
```

3.5 Tiled data analysis

`methylSig` package also provides `methylSigTile()` function to tile data within continuous non-overlapping windows. The default window size is 25bp. After tiling data, the `methylSigCalc()` function can be used to calculate differential methylation statistics.

```
### Tiled analysis
methTile = methylSigTile(meth,win.size = 25)
myDiffSigbothTile = methylSigCalc(methTile, groups=c(1,0), min.per.group=3)
```

```
## Total number of regions: 994
```

3.6 Variance from one group

Using the `dispersion` argument, it is possible to estimate variances from one group rather than from both groups. The following code calculates differential methylation statistics based on estimating variances from group 0 only.

```
### Variance from sample treatment group "0" only
myDiffSignorm = methylSigCalc(meth, groups=c(1,0), dispersion=0, min.per.group=3)
```

```
## Total number of bases: 3.26k
```

```
myDiffSignormTile = methylSigCalc(methTile, groups=c(1,0),
                                  dispersion=0, min.per.group=3)
```

```
## Total number of regions: 994
```

3.7 Using local information

It is also possible to use information from nearby CpG sites to improve the variance and methylation level estimates. The default `winsize.disp` and `winsize.meth` are 200 bps. The `winsize.disp` argument only takes into effect when `local.disp` is set to `TRUE'. Similarly \verb@winsize.meth@ argument only takes into effect when \verb@local.meth@ is set to`TRUE'.

```
### Variance from both groups and using local information for variance
myDiffSigBothLoc = methylSigCalc(meth, groups=c(1,0),
          min.per.group=3, local.disp=TRUE, winsize.disp=200)
```

```
## Total number of bases: 3.26k
```

```
### Variance from sample treatment group "0" only and using local information for variance
myDiffSignormLoc = methylSigCalc(meth, groups=c(1,0), dispersion=0,
          min.per.group=3, local.disp=TRUE, winsize.disp=200)
```

```
## Total number of bases: 3.26k
```

```
### Variance from both groups and using local information for methylation level
myDiffSigBothMLoc = methylSigCalc(meth, groups=c(1,0),
          min.per.group=3, local.meth=TRUE, winsize.meth=200)
```

```
## Total number of bases: 3.26k
```

```
### Variance from both groups and using local information for methylation level and variance
myDiffSigBothMDLoc = methylSigCalc(meth, groups=c(1,0),
          min.per.group=3, local.disp=TRUE, winsize.disp=200,
          local.meth=TRUE, winsize.meth=200)
```

```
## Total number of bases: 3.26k
```

3.8 Multi-thread computation

`methylSig` provides multicore programming to substantially reduce data analysis time. In the functions `methylSigReadData` and `methylSigCalc`, multi-core programming will be initiated using `num.cores` argument. Note that this option depends on R package `parallel` and hence is not available in the Windows platform. The following example illustrates the use of 2 cores.

```
#### Read Data using 2 cores
meth <- methylSigReadData(fileList, sample.ids = sample.id, assembly = "hg18",
           treatment = treatment, context = "CpG", destranded=TRUE,
           num.cores=2, quiet=TRUE)

#### Differential methylation analysis using 2 cores
myDiffSigboth = methylSigCalc(meth, groups=c(1,0), min.per.group=3, num.cores=2)
```

4 Annotation

4.1 CpG islands

There are two functions, `cpgAnnotation()` and `cpgAnnotationPlot()`, in the `methylSig` package for CpG island annotation. The CpG island information file can be download the UCSC genome browser. The appropriate genome assembly should be used.

In Linux, the user may use the following command to download the annotation file for hg19. Please use appropriate directories for hg18, mm9 or mm10.

```
wget ftp://hgdownload.cse.ucsc.edu/goldenPath/hg19/database/cpgIslandExt.txt.gz
gunzip *.gz
```

Here we use the CpG island annotation file provided in the `methylSig` package to annotate our example. Note that this is a reduced annotation file and is not appropriate for a full real data analysis.

```
library("graphics")
cpgInfo = getCpGInfo(system.file("annotation", "cpgi.hg18.bed.txt",
                                 package = "methylSig"))

myDiffDMCs = myDiffSigboth[myDiffSigboth[,"qvalue"] < 0.05
                           & abs(myDiffSigboth[,"meth.diff"])>25,]
cpgAnn = cpgAnnotation(cpgInfo,myDiffSigboth)
cpgAnnDmc = cpgAnnotation(cpgInfo, myDiffDMCs)
cpgAnnotationPlot(cpgAnn,main="ALL")
```

```
cpgAnnotationPlot(cpgAnnDmc,main="DMCs")
```

4.2 RefGene annotation

Again, there are two functions, `refGeneAnnotation()` and `refGeneAnnotationPlot()`, in `methylSig` package for annotation using RefGene models. The refGene information file can be download from websites such

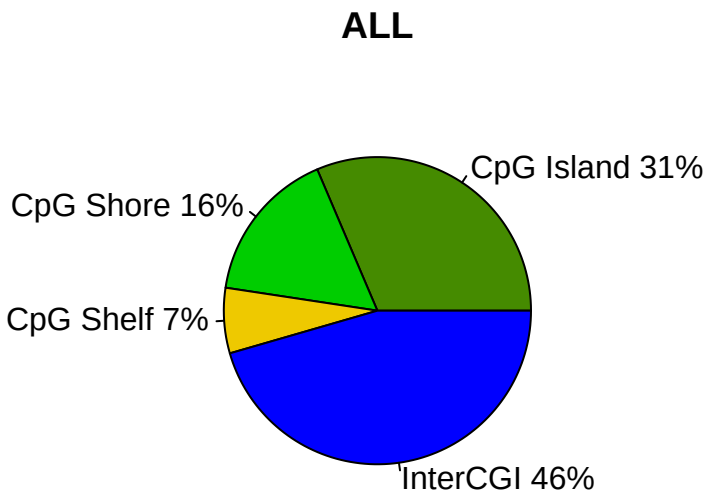

Figure 1: CpG annotation plots

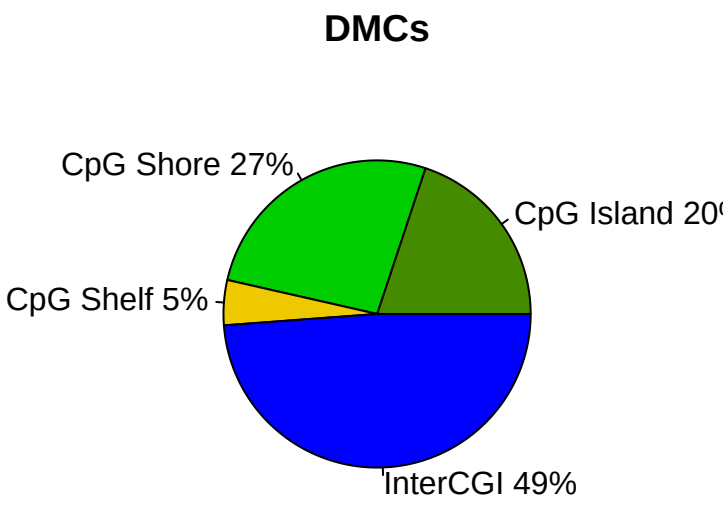

Figure 2: CpG annotation plots

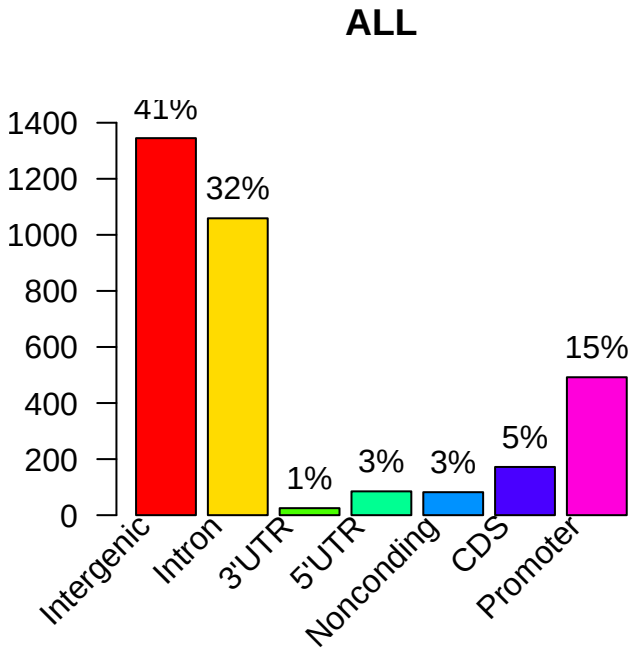

Figure 3: refGene annotation plots

UCSC genome browser. The appropriate genome assembly (the same genome assembly of the provided data) should be used.

In a linux server, the user may use the following command to download the annotation file for hg19. Please use appropriate directories for hg18, mm9 or mm10.

```
wget ftp://hgdownload.cse.ucsc.edu/goldenPath/hg19/database/refGene.txt.gz
gunzip *.gz
```

We use refGene annotation file provided in the `methylSig` package to annotate in our example. Note that this is a reduced annotation file and is not appropriate for the a full real data analysis.

```
refGeneInfo = getRefgeneInfo(system.file("annotation", "refGene.txt",
                                  package = "methylSig"))

refGeneAnn = refGeneAnnotation(refGeneInfo, myDiffSigboth)
refGeneAnnDmc = refGeneAnnotation(refGeneInfo, myDiffDMCs)
refGeneAnnotationPlot(refGeneAnn,main="ALL",
                priority=c("promoter","cds", "noncoding", "5'utr", "3'utr"))
```

```
refGeneAnnotationPlot(refGeneAnnDmc, main="DMC",
                priority=c("promoter","cds", "noncoding", "5'utr", "3'utr"))
```

4.3 Transcription factor (TF) enrichment test

The functions `getTFBSInfo()` and `methylSig.tfbsEnrichTest()` are provided for reading the TFBS information file and implementing transcription factor enrichment test.

UCSC genome browser provides TFBS conserved track for hg18 and hg19. The following linux server shell command can be used to download these files:

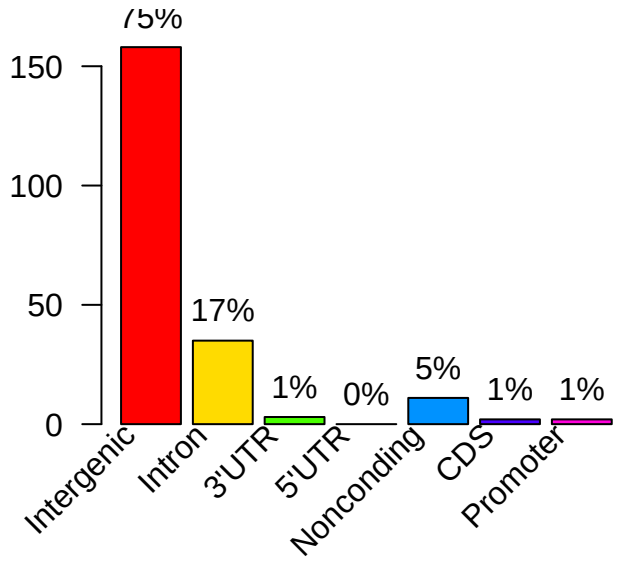

Figure 4: refGene annotation plots

```
wget ftp://hgdownload.cse.ucsc.edu/goldenPath/hg19/database/tfbsConsSites.txt.gz
wget ftp://hgdownload.cse.ucsc.edu/goldenPath/hg19/database/tfbsConsFactors.txt.gz
gunzip *.gz
```

Here, `tfbsConsSites.txt` is tracking information and can be used directory in function `getTFBSInfo()`. The explanation of variable names is listed in file `tfbsConsFactors.txt`.

Another TFBS track is from ENCODE for hg18, hg19 and mm9. However, the `methylSig` package cannot use this type of track directly. We provide ENCODE TFBS track files that suitable for `methylSig` package at http://sartorlab.ccmb.med.umich.edu/software.

```
tfbsInfo = getTFBSInfo(system.file("annotation", "tfbsUniform.txt",
                                   package = "methylSig"))
DMCIndex = (myDiffSigboth[,"qvalue"] < 0.05
                & abs(myDiffSigboth[,"meth.diff"]) > 25)
pvalue = methylSig.tfbsEnrichTest(myDiffSigboth, DMCIndex, tfbsInfo)
```

To identify which TFs have significant level of hypermethylation or hypomethylation across their binding sites, which could indicate whether the TF is having a weaker or stronger regulatory effect, respectively, we first tile all reads from regions to which a particular TF is predicted to bind. We then apply our beta-binomial model to the data for each TF to identify TFs with hyper- or hypo-methylated binding sites.

To achieve this, we provide function `methylSigTileTFBS()` to tile all data corresponding to the same TF.

```
methTileTFs = methylSigTileTFBS(meth, tfbsInfo)
myDiffTFs = methylSigCalc(methTileTFs, groups=c(1,0))
```

```
## Total number of TFs: 126
```

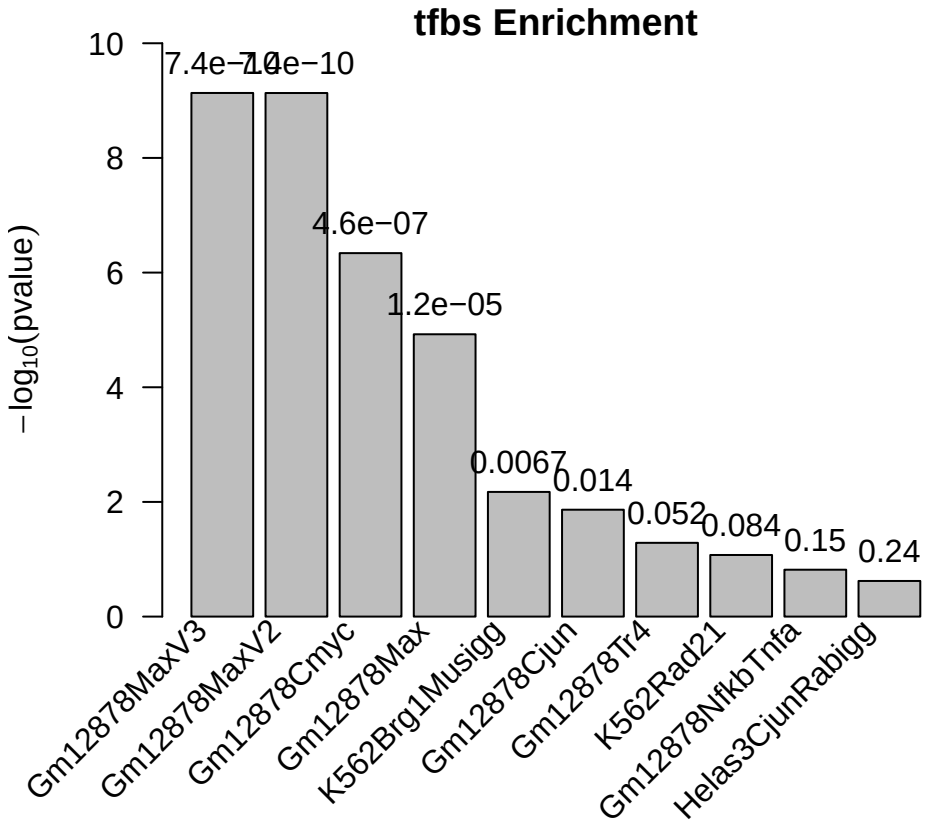

Figure 5: TFBS Enrichment

4.4 Data visualization

MethylSig offers a unique two-tiered visualization of the methylation data depending on the zoom level. When the chromosome range is large (>1 million bp), the visualization function does not show individual sample data.

```
methylSigPlot(meth, "chr21", c(43000000, 43500000), groups=c(1,0),
    cpgInfo=cpgInfo,refGeneInfo=refGeneInfo,
    myDiff=myDiffSigboth,tfbsInfo=tfbsInfo,tfbsDense=F,sigQ=0.05)
```

```
## Warning in methylSigPlot(meth, "chr21", c(4.3e+07, 43500000), groups =
## c(1, : Range of the drawing area is two wide, please reduce the range!
```

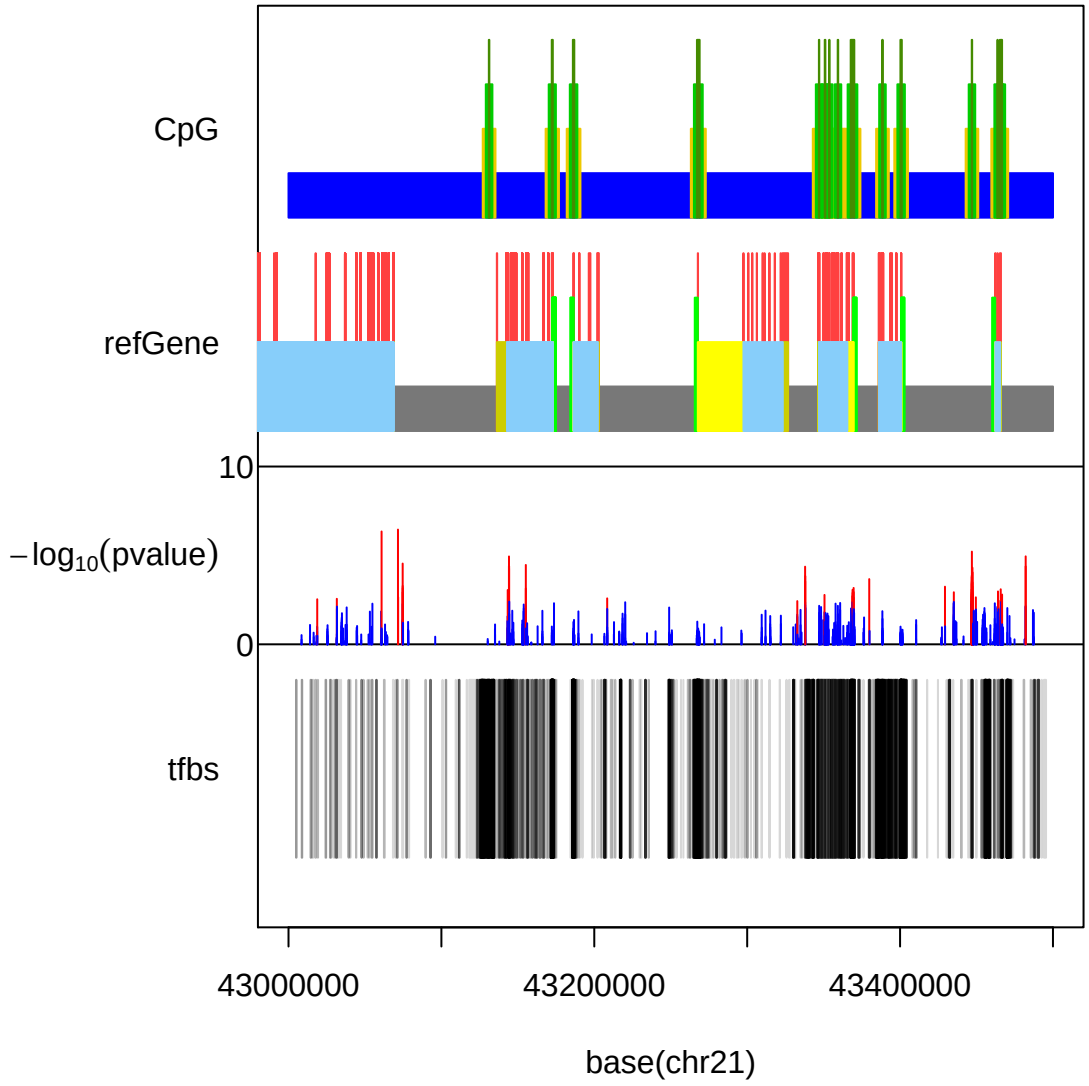

Figure 6: Data visualization over a large range

For narrow regions where at most 500 CpG sites have data reads, users can visualize sample-specific coverage levels and % methylation at each site, together with group averages, significance levels and a number of genomic annotations.

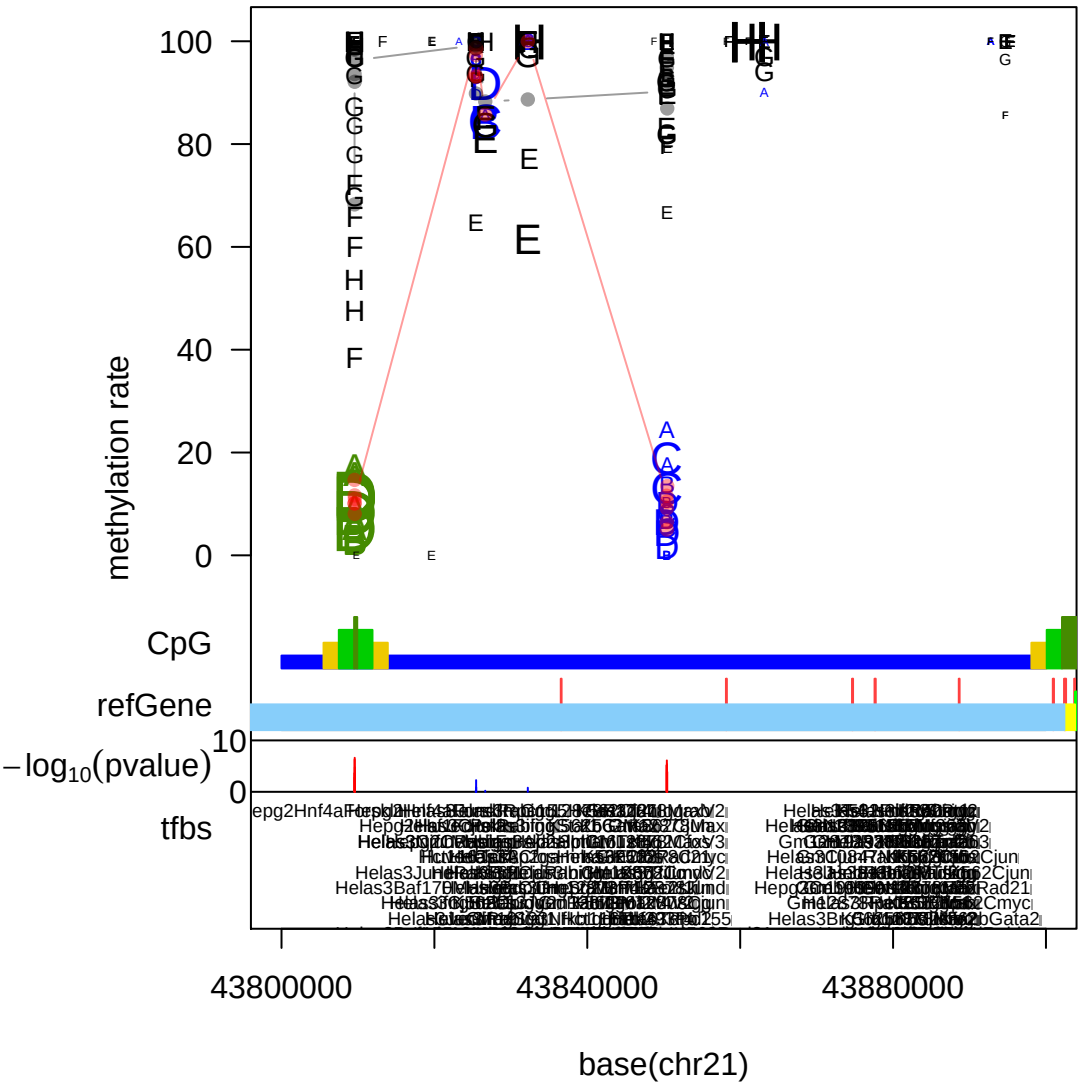

Figure 7: Data visualization over a narrow range

```
methylSigPlot(meth, "chr21", c(43800000, 43900000), groups=c(1,0),
    cpgInfo=cpgInfo, refGeneInfo=refGeneInfo,
    myDiff=myDiffSigboth,tfbsInfo=tfbsInfo,tfbsDense=F,sigQ=0.05)
```

5 Data classes

5.1 methylSigData object

5.1.1 S4 data structure

The `methylSig` package uses S4 object. The contents of `methylSigData@` can be shown using the `show()` function in R or just type the object itself.

```
show(meth)
```

```
## methylSigData object with 7,571 rows
## --------------------------
##      chr    start      end strand coverage1 numCs1 numTs1 coverage2 numCs2
## 1  chr21 43000564 43000564      -        NA     NA     NA        NA     NA
## 2  chr21 43000820 43000820      -        NA     NA     NA        NA     NA
## 3  chr21 43008527 43008527      +        32     32      0       124    122
## 4  chr21 43008531 43008531      +        32     31      1       125    125
## 5  chr21 43008543 43008543      +        32     29      3       125    117
## 6  chr21 43008665 43008665      -        NA     NA     NA        NA     NA
## 7  chr21 43008673 43008673      -        27     27      0        29     25
## 8  chr21 43008709 43008709      -        67     63      4        31     26
## 9  chr21 43008719 43008719      -        67     62      5        33     33
## 10 chr21 43014041 43014041      +        NA     NA     NA       207    202
##    numTs2 coverage3 numCs3 numTs3 coverage4 numCs4 numTs4 coverage5 numCs5
## 1      NA        NA     NA     NA        13     13      0        NA     NA
## 2      NA        NA     NA     NA        NA     NA     NA        NA     NA
## 3       2        48     44      4        NA     NA     NA        95     95
## 4       0        48     48      0        NA     NA     NA        94     94
## 5       8        48     44      4        NA     NA     NA        94     81
## 6      NA        NA     NA     NA        NA     NA     NA       148    148
## 7       4        NA     NA     NA        NA     NA     NA       146    146
## 8       5        NA     NA     NA        NA     NA     NA       149    149
## 9       0        NA     NA     NA        NA     NA     NA       149    149
## 10      5       190    184      6       228    228      0       136    132
##    numTs5 coverage6 numCs6 numTs6 coverage7 numCs7 numTs7 coverage8 numCs8
## 1      NA        NA     NA     NA        NA     NA     NA        NA     NA
## 2      NA        NA     NA     NA        11     11      0        NA     NA
## 3       0        93     93      0       108    104      4        NA     NA
## 4       0        93     93      0       104     95      9        NA     NA
## 5      13        93     93      0       105     96      9        NA     NA
## 6       0       178    178      0       162    156      6        73     73
## 7       0       178    156     22       162    161      1        73     73
## 8       0       184    184      0       163    156      7        74     74
## 9       0        NA     NA     NA       165    158      7        74     74
## 10      4       155    135     20       203    198      5        61     61
##    numTs8
## 1      NA
## 2      NA
## 3      NA
## 4      NA
## 5      NA
## 6       0
## 7       0
## 8       0
## 9       0
## 10      0
## --------------------------
## sample.ids: AML1 AML2 AML3 AML4 NBM1 NBM2 NBM3 NBM4
## treatment: 1 1 1 1 0 0 0 0
## destranded: TRUE
## resolution: base
```

```
## options: maxCount=500 & minCount=10 & filterSNPs=FALSE & assembly=hg18 & context=CpG
```

Here, NA means there was no data at this base location on the related sample.

5.1.2 Subsetting

Data can be subset using matrix style operations. Row represents base location and each column is a sample. Below is an example to obtain data for samples 1 to 4:

```
meth1_4 = meth[,1:4]
```

This example returns the first 100 methylation reads in the data:

```
methSub1_100 = meth[1:100,]
```

Two arguments can be used together. This example returns the first 100 methylation reads for samples 1 and 2.

```
methSubData = meth[1:100,1:2]
methSubData
```

```
## methylSigData object with 60 rows
## --------------------------
##      chr    start      end strand coverage1 numCs1 numTs1 coverage2 numCs2
## 1  chr21 43008527 43008527      +        32     32      0       124    122
## 2  chr21 43008531 43008531      +        32     31      1       125    125
## 3  chr21 43008543 43008543      +        32     29      3       125    117
## 4  chr21 43008673 43008673      -        27     27      0        29     25
## 5  chr21 43008709 43008709      -        67     63      4        31     26
## 6  chr21 43008719 43008719      -        67     62      5        33     33
## 7  chr21 43014041 43014041      +        NA     NA     NA       207    202
## 8  chr21 43014044 43014044      +        NA     NA     NA       207    195
## 9  chr21 43014076 43014076      +        NA     NA     NA       202    188
## 10 chr21 43016439 43016439      +        55     51      4        93     90
##    numTs2
## 1       2
## 2       0
## 3       8
## 4       4
## 5       5
## 6       0
## 7       5
## 8      12
## 9      14
## 10      3
## --------------------------
## sample.ids: AML1 AML2
## treatment: 1 1
## destranded: TRUE
## resolution: base
## options: maxCount=500 & minCount=10 & filterSNPs=FALSE & assembly=hg18 & context=CpG
```

5.1.3 Getting values

If the second argument is a string that matches one of the column names in the `methylSigData` object, it gives the values of that column. Valid column names are `chr`, `start`, `end`, `strand`, `coverage1`, ..., `numCs1`, ..., and `numTs1`.

```
coverage1 = meth[,"coverage1"]
startTop200 = meth[1:200,"start"]
```

5.2 methylSigDiff object

5.2.1 S4 data structure

The contents of `methylSigDiff` are:

```
myDiffSigboth
```

```
## methylSigDiff object with 3,260 rows
## --------------------------
##      chr    start      end strand     pvalue    qvalue  meth.diff
## 1  chr21 43008527 43008527      + 0.30197133 0.6323745 -2.4167475
## 2  chr21 43008531 43008531      + 0.86910547 1.0000000  0.4158129
## 3  chr21 43008543 43008543      + 0.54717680 0.8571823 -2.8071020
## 4  chr21 43014041 43014041      + 0.35454483 0.6814532  2.3585424
## 5  chr21 43014044 43014044      + 0.91546697 1.0000000 -0.6133413
## 6  chr21 43014076 43014076      + 0.08082989 0.3398239 12.1681339
## 7  chr21 43016439 43016439      + 0.22017979 0.5537706 -2.1713344
## 8  chr21 43016517 43016517      - 0.38867776 0.7094566 -2.0930416
## 9  chr21 43018213 43018213      + 0.83266057 1.0000000 -1.4186090
## 10 chr21 43018274 43018274      - 0.32935865 0.6514688  2.0009532
##    logLikRatio        theta df      mu1      mu0
## 1   1.27484057 3.689198e+01  6 96.49842 98.91517
## 2   0.02957797 1.172001e+01  6 98.45532 98.03950
## 3   0.40674783 3.131946e+01  6 90.97520 93.78231
## 4   0.98274395 2.829499e+01  7 97.82788 95.46934
## 5   0.01210852 1.269081e+01  7 90.45700 91.07034
## 6   4.39674527 1.904543e+01  6 93.23638 81.06825
## 7   1.76894807 9.151486e+01  8 96.96983 99.14116
## 8   0.86326517 1.000000e+06  6 93.10351 95.19656
## 9   0.04765519 7.094155e+00  8 91.07918 92.49778
## 10  1.09884498 7.115526e+01  7 98.16913 96.16818
## --------------------------
## sample.ids: AML1 AML2 AML3 AML4 NBM1 NBM2 NBM3 NBM4
## treatment: 1 1 1 1 0 0 0 0
## destranded: TRUE
## resolution: base
## options: dispersion=both & local.disp=FALSE & local.meth=FALSE& min.per.group=c(3,3)& Total: 3260
```

5.2.2 Subsetting

This object can also subset by row to obtain results from part of CpG sites or regions. However, the qvalues will not be readjusted.

```
myDiff100 = myDiffSigboth[1:100,]
```

5.2.3 Getting values

Similar to the `methSigData` object, if the second argument is a string that is the same as one of the column names, it will return the results for that column. The valid variable names are `chr`, `start`, `end`, `strand`, `pvalue`, `qvalue`, `meth.diff`, `logLikRatio`, `theta`, `df`, `mu1`, and `mu0`. Here, for group methylation mean estimates `mu1` and `mu0`, 1 and 0 come from the `groups` argument in the `methylSigCalc()` function. So if one has run the `methylSigCalc()` function with `groups=c(4,0)`, then `mu4` and `mu0` will appear in the results.

```
qvalues = myDiffSigboth[,"qvalue"]
```

5.2.4 How to subtract DMCs or DMRs

This `methylSigDiff` object is very flexible to use by combining functions of subsetting and getting values. For example, the following code can obtain differentially methylated cytosines or regions defined as qvalue $<$ 0.05 and difference of methylation rate $>$ 25%.

```
myDiffq05D25 = myDiffSigboth[myDiffSigboth[,"qvalue"] < 0.05
                             & abs(myDiffSigboth[,"meth.diff"]) > 25,]
```

Here `abs()` is the absolute value function in R.

If you want to use pvalues instead of qvalues, then you can use

```
myDiffp05D25 = myDiffSigboth[myDiffSigboth[,"pvalue"] < 0.05
                             & abs(myDiffSigboth[,"meth.diff"]) > 25,]
```

5.3 Summarizing data

You can easily use other R functions to summarize or draw plots.

```
methRaw = methylSigReadData(fileList, sample.ids = sample.id,assembly = "hg18",
             treatment = treatment, context = "CpG", minCount = 0,
             maxCount=Inf, destranded=F, quiet=T)
```

```
## (1/8) Count Invalid List: 0/2411=0
## (2/8) Count Invalid List: 0/4259=0
## (3/8) Count Invalid List: 0/3861=0
## (4/8) Count Invalid List: 0/3750=0
## (5/8) Count Invalid List: 0/6228=0
## (6/8) Count Invalid List: 0/6430=0
## (7/8) Count Invalid List: 0/5962=0
## (8/8) Count Invalid List: 0/3694=0
```

```
summary(methRaw[,"numCs1"]/methRaw[,"coverage1"])
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   0.000   0.069   0.861   0.612   1.000   1.000    3606
```

```
summary(methRaw[,"coverage1"])
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   10.00   14.00   23.00   30.68   39.00  177.00    3606
```

```
hist(methRaw[,"numCs1"]/methRaw[,"coverage1"],
                main="Histogram of methylation rate for sample 1",
                xlab="methylation rate")
```

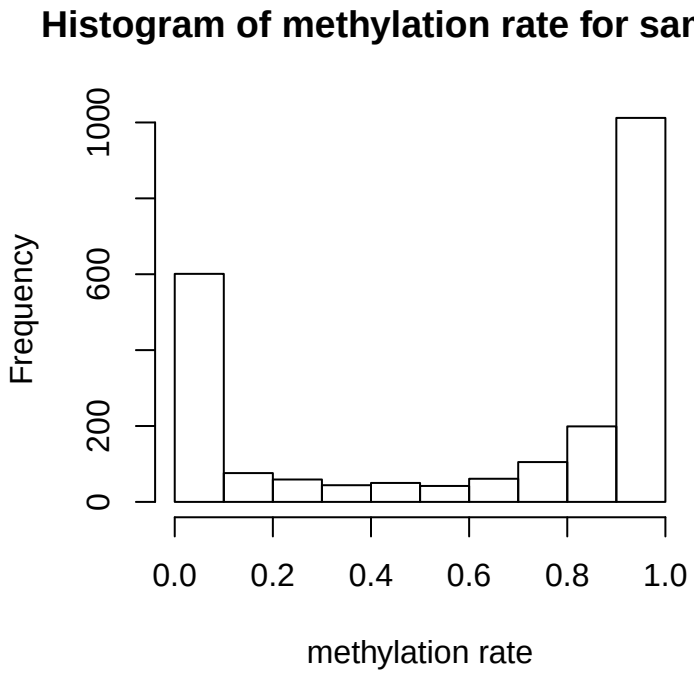

Figure 8: Methylation rate for sample 1

```
hist(methRaw[,"coverage1"], main="Histogram of coverage for sample 1",
                xlab="coverage")
```

5.4 Generating heatmaps

Here we provide an example to generate a correlation heatmap.

```
library(gplots)

x = meth[,"numCs"]/meth[, "coverage"]
colnames(x) = meth@sample.ids
rownames(x) =rep(NA, NROW(x))

corrALL = cor(x, use="pairwise.complete.obs")
heatmap.2(1-corrALL, na.rm=T, breaks=100,
          hclustfun = function(x) hclust(x,method="ward.D"),
          col="bluered",trace="none", symm=T, keysize=1,density.info="none")
```

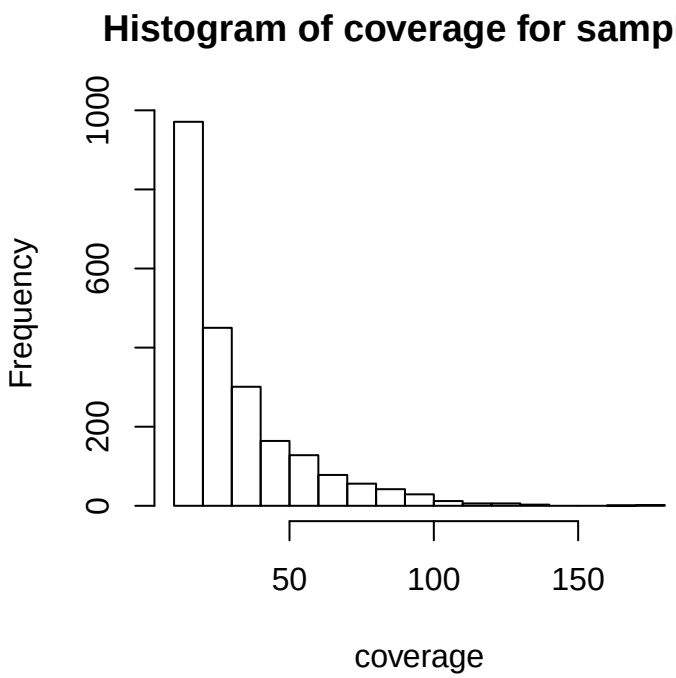

Figure 9: Coverage for sample 1

Here is another example to generate a correlation heatmap based on differentially methylated cytosines.

```
myDiffDMC = myDiffSigboth[myDiffSigboth[,"qvalue"] < 0.05
                          & abs(myDiffSigboth[,"meth.diff"]) >=25,]
listInMeth = match(myDiffDMC@data.ids, meth@data.ids)
y = x[listInMeth,]
corrDMC = cor(y, use="pairwise.complete.obs")
heatmap.2(1-corrDMC, na.rm=T, breaks=100,
          hclustfun = function(x) hclust(x,method="ward.D"),
          col="bluered",trace="none", symm=T, keysize=1,density.info="none")
```

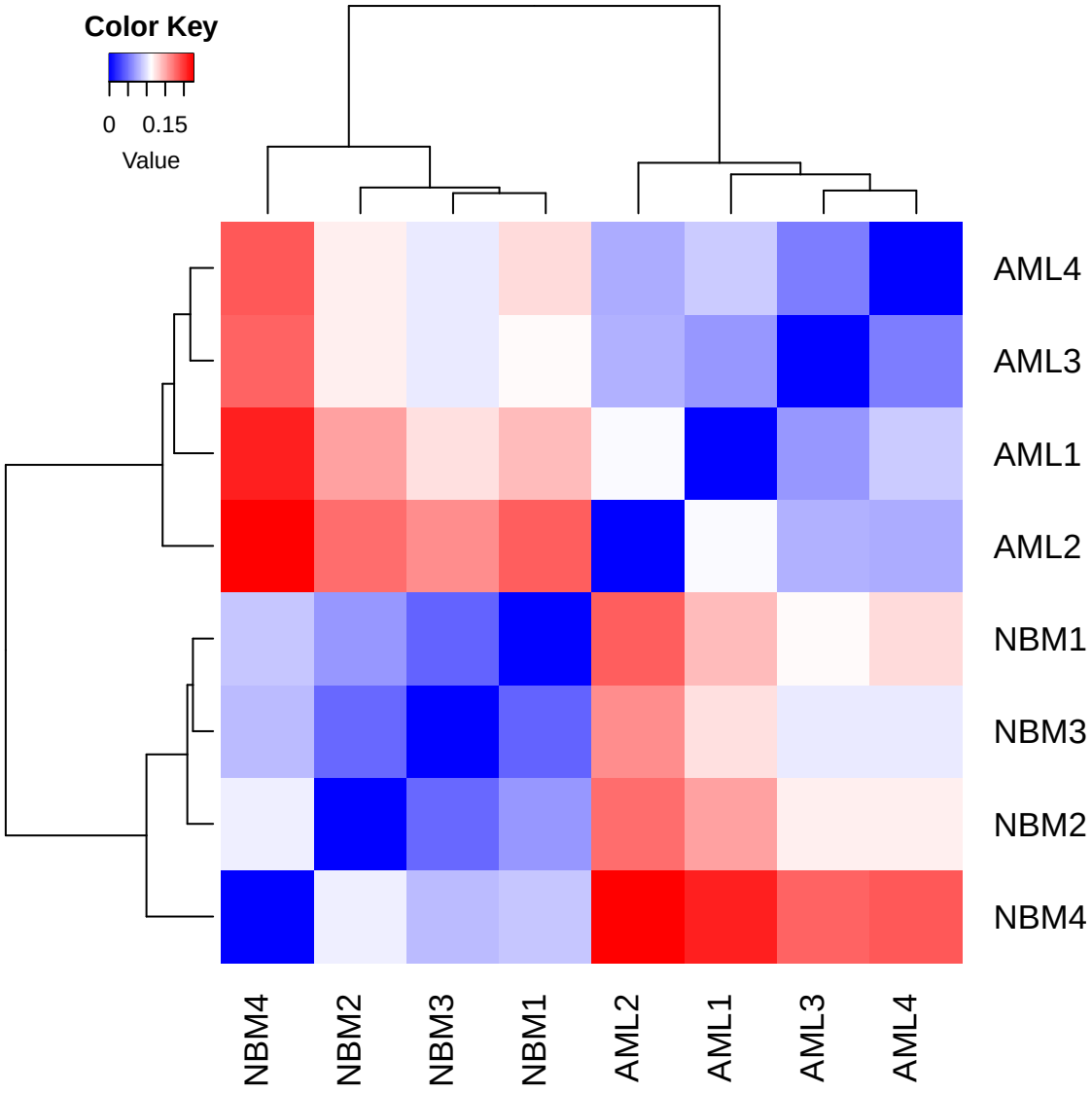

Figure 10: Heatmap based on methylation rate at all CpG sites

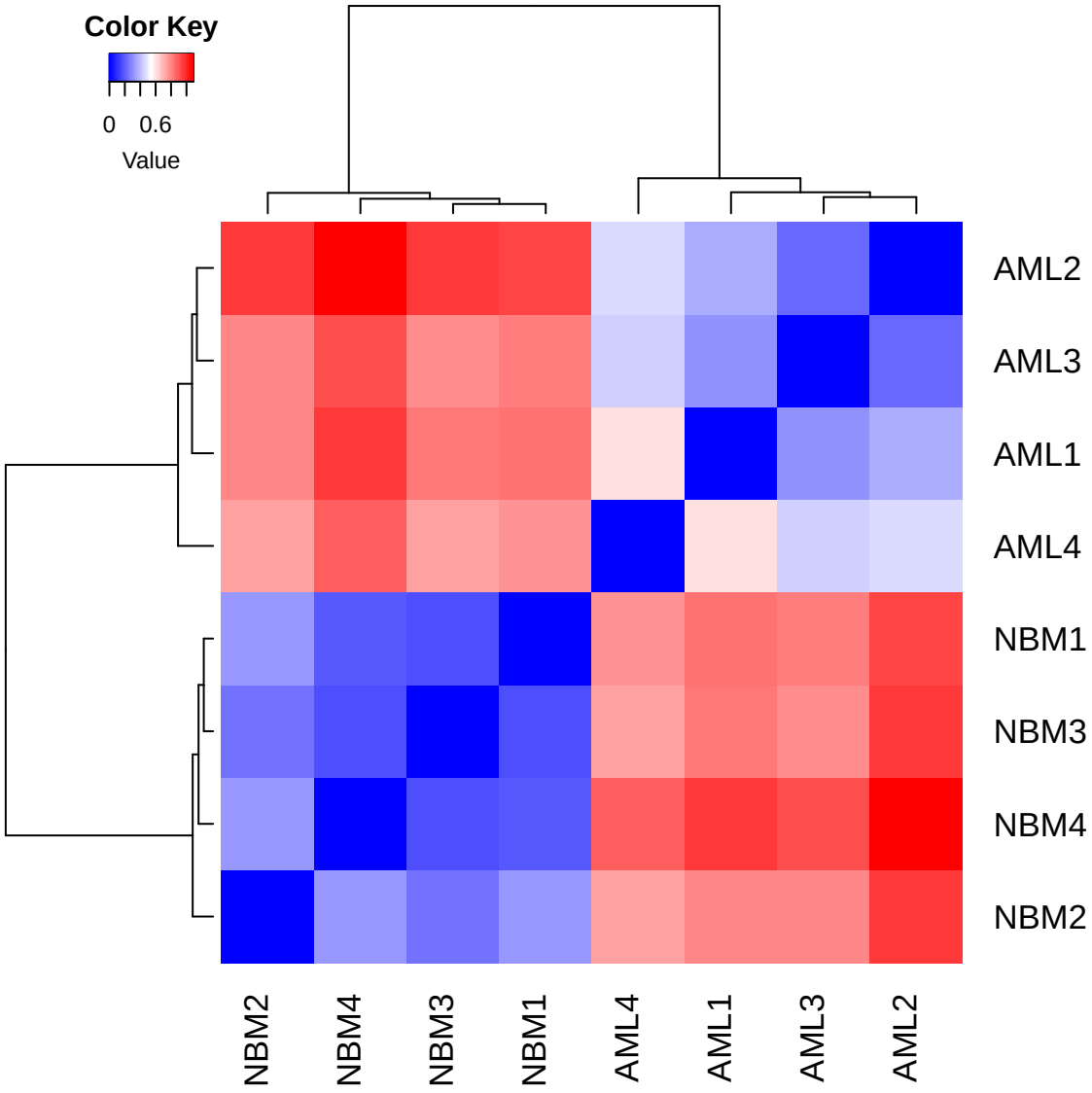

Figure 11: Heatmap based on methylation rate at differentially methylated sites