

MEDIPS Tutorial

Lukas Chavez

May 26, 2010

1 Introduction

MEDIPS was developed for analyzing data derived from methylated DNA immunoprecipitation (MeDIP) experiments [Weber et al., 2005] followed by sequencing (MeDIP-Seq). Nevertheless, functionalities like the saturation analysis may be applied to other types of sequencing data (e.g. ChIP-Seq). MEDIPS addresses several aspects in the context of MeDIP-Seq data analysis. These are:

- estimating the reproducibility for obtaining full genome methylation profiles with respect to the total number of given short reads and to the size of the reference genome,
- analyzing the coverage of genome wide DNA sequence patterns (e.g. CpGs) by the given reads,
- calculating an CpG enrichment factor as a quality control for the immunoprecipitation,
- calculating genome wide MeDIP-Seq signal densities at a user specified resolution,
- calculating genome wide sequence pattern densities (e.g. CpGs) at a user specified resolution,
- plotting of calibration plots as a data quality check and for a visual inspection of the dependency between local sequence pattern (e.g. CpG) densities and MeDIP signals
- normalization of MeDIP-Seq data with respect to local sequence pattern (e.g. CpG) densities,
- summarized methylation values for genome wide windows of a specified length or for user supplied regions of interest (ROIs),
- calculating differentially methylated regions on raw or normalized data comparing two sets of MeDIP-Seq data,
- export of raw and normalized data for visualization in common genome browsers (e.g. the UCSC genome browser).

MEDIPS starts directly where the mapping tools stop and can be used for any genome of interest, limited only by the available genomes within Bioconductors `BSgenome` package.

2 Preparations

In order to execute MEDIPS, you need to have some other packages installed in your R library. These are **BSgenome** and **gtools**, as well as the packages they depend on. You can check this, by starting R and typing

```
> packageDescription("BSgenome")
> packageDescription("gtools")
```

R will inform you in case you do not have these packages installed. In case you do not have these necessary packages installed, start R and try typing

```
> source("http://bioconductor.org/biocLite.R")
> biocLite("BSgenome")
> biocLite("gtools")
```

Please be aware of having the latest BSgenome version installed (>1.14.2). The version number is returned by the `packageDescription()` command. Please note, unfortunately, there is a bug in the R distributions versions ≤2.10.1 that sometimes causes errors when interacting with the BSgenome package. This bug should be fixed in R versions >2.11.0. Therefore, it is recommended to update your R environment to the latest version.

Next, it is necessary to install the MEDIPS package into your R environment. Currently, the MEDIPS package `MEDIPS_1.2.tar.gz` can be downloaded from <http://medips.molgen.mpg.de/>. In order to install MEDIPS, download the package to your R working environment and type

```
> install.packages("MEDIPS_1.2.tar.gz", dependencies = TRUE, repos = NULL)
```

In case, the downloaded MEDIPS package is not in your current working directory, you have to specify the file name together with the full path. In order to reproduce the presented MEDIPS workflow, we recommend to download the example data sets `MeDIP_hESCs_chr1-3.txt.gz` (35M), `MeDIP_DE_chr1-3.txt.gz` (42M), and `Input_StemCells_chr1-3.txt.gz` (34M) from <http://medips.molgen.mpg.de/> and gunzip the files. The files contain genomic regions from chromosomes 1, 2, and 3 only, as covered by short reads obtained from a MeDIP experiment of human embryonic stem cells (hESCs), a MeDIP experiment of differentiated hESCs (definitive endoderm, DE), and of INPUT experiments [Chavez et al., 2010].

As input, MEDIPS requires tab-separated files without headers containing four columns:

- the first column is of type character and contains the chromosome of the region (e.g. chr1).
- the second column is of type numeric and contains the start position of the mapped read.
- the third column is of type numeric and contains the stop position of the mapped read.
- the fourth column is of type character and contains the strand information of the mapped read.

Each row represents a mapped read. These informations can be extracted from the output file(s) of common mapping tools. MEDIPS counts chromosome sequence positions starting at 1. Some alignment tools output the mapped regions by interpreting the first base of a chromosome as 0. MEDIPS requires one input file for each condition that has to be analyzed. Region informations from several mapping results have to be pooled into one file. Furthermore, it might be worthwhile to filter out some mapped reads of low quality or to exclude artificial short read pile-ups from the results of the mapping procedure. Please note, any such data pooling, further filtering or correction for the start and stop positions has to be done by yourself before using MEDIPS.

Next, you need to have your genome of interest available. As soon as you have the BSgenome package installed and the library loaded using

```
> library("BSgenome")
```

you can list all available genomes by typing

```
> available.genomes()

[1] "BSgenome.Amelifera.BeeBase.assembly4"
[2] "BSgenome.Amelifera.UCSC.apiMel2"
[3] "BSgenome.Athaliana.TAIR.01222004"
[4] "BSgenome.Athaliana.TAIR.04232008"
[5] "BSgenome.Btaurus.UCSC.bosTau3"
[6] "BSgenome.Btaurus.UCSC.bosTau4"
[7] "BSgenome.Celegans.UCSC.ce2"
[8] "BSgenome.Cfamiliaris.UCSC.canFam2"
[9] "BSgenome.Dmelanogaster.UCSC.dm2"
[10] "BSgenome.Dmelanogaster.UCSC.dm3"
[11] "BSgenome.Drerio.UCSC.danRer5"
[12] "BSgenome.Ecoli.NCBI.20080805"
[13] "BSgenome.Ggallus.UCSC.galGal3"
[14] "BSgenome.Hsapiens.UCSC.hg17"
[15] "BSgenome.Hsapiens.UCSC.hg18"
[16] "BSgenome.Hsapiens.UCSC.hg19"
[17] "BSgenome.Mmusculus.UCSC.mm8"
[18] "BSgenome.Mmusculus.UCSC.mm9"
[19] "BSgenome.Ptroglodytes.UCSC.panTro2"
[20] "BSgenome.Rnorvegicus.UCSC.rn4"
[21] "BSgenome.Scerevisiae.UCSC.sacCer1"
[22] "BSgenome.Scerevisiae.UCSC.sacCer2"
```

In the given example, we mapped the short reads against the human genome build hg19. Therefore, we download and install this genome build:

```
> source("http://bioconductor.org/biocLite.R")
> biocLite("BSgenome.Hsapiens.UCSC.hg19")
```

This takes some time, but has to be done only once for each necessary reference genome.

3 MEDIPS Workflow

3.1 Create a MEDIPS SET from the input file

First, you have to load MEDIPS into your R environment. Herewith, the dependent libraries `BSgenome` and `gtools` as well as the packages they depend on will be loaded, additionally.

```
> library(MEDIPS)
```

In the given example, we mapped the short reads against the human genome build hg19. Therefore, we load the pre-installed (see chapter 2) hg19 library:

```
> library(BSgenome.Hsapiens.UCSC.hg19)
```

Next, a MEDIPS SET is created by reading the input file. For this, you have to state the reference genome build and your input file (in this example: `MeDIP_hESCs_chr1-3.txt`). You can also state the full directory path together with the file name. In case you know the number of rows within your input file, you can specify the `numrows` parameter. This will accelerate the reading of your file. Otherwise just skip the `numrows` parameter. In our example file, there are 5,461,263 Mio regions within the input file (here, only chromosomes 1, 2, and 3 are included).

```
> CONTROL.SET = MEDIPS.readAlignedSequences(BSgenome = "BSgenome.Hsapiens.UCSC.hg19",  
+      file = "MeDIP_hESCs_chr1-3.txt", numrows = 5461263)
```

MEDIPS now created a MEDIPS SET from the input. The current content of a MEDIPS SET can be viewed at any time by typing the name of your MEDIPS SET object.

```
> CONTROL.SET
```

```
S4 Object of class MEDIPSset
=====
Regions informations
=====
Regions file:  MeDIP_hESCs_chr1-3.txt
Organism:  BSgenome.Hsapiens.UCSC.hg19
Chromosomes: chr1 chr2 chr3
Chromosome lengths: 249250621 243199373 198022430
Number of regions: 5461263
Regions chromosomes: chr1 chr1 chr1...
Regions start positions: 100001075 10000155 10000178...
Regions stop positions: 100001111 10000191 10000214...
Regions strand: - - -...
=====
Genome vector signals informations
=====
Genome wide bin size:
Reads extended by:
Genome vector chromosomes: NA NA NA...
```

```

Genome vector positions: NA NA NA...
Genome vector signals: NA NA NA...
=====
Pattern informations
=====
Pattern:
Number of patterns:
Pattern chromosomes: NA NA NA...
Pattern positions: NA NA NA...
=====
Genome vector coupling factor informations
=====
Distance function:
Distance file:
Fragment length:
Genome vector coupling factors: NA...
=====
Calibration informations
=====
Calibration curve mean signals: NA...
Calibration curve mean coupling factors: NA...
Calibration curve variance: NA...
Intercept:
Slope:
Calibration chromosome:
=====
Genome vector normalized signal informations
=====
Normalization output interval: [0:1000]
Genome vector normalized signals: NA...

```

After reading the input file, the MEDIPS SET contains only the information about the input regions, like the input file name, the dependent organism, the chromosomes included in the input file, the length of the included chromosomes (automatically loaded), the number of regions, and the start, stop and strand informations of the regions. All further slots, for example for the weighting parameters and normalized data are still empty and will be filled during the workflow.

3.2 Creating the Genome Vector and Export of RPM Signals

Based on the given regions, a genome-wide coverage has to be calculated. In order to calculate the genome wide short read coverage, the user has to specify a targeted data resolution using the parameter `bin_size` (default: 50bp). In principle, a `bin_size=1` can be specified. Because the resolution of MeDIP-Seq data is restricted by the size of the sonicated DNA fragments after amplification and size selection (that often is between 0.2-1kb), it might not be necessary to specify very small bin sizes. Moreover, the smaller the bin size, the higher the need for memory of your computer and the higher the runtime will be. We

consider a bin size of 50bp as a reasonable compromise on data resolution and computational costs.

Each chromosome inside the MEDIPS SET will then be divided into bins of size 50bp and the short read coverage will be calculated on this resolution. In the following, we call the bin representation of the genome the genome vector.

Moreover, short reads generated by modern-day sequencers do not represent the full DNA fragments but are of shorter length (e.g. 36bp). Therefore, a smoothing of the data is recommended by extending the reads. This can be achieved by setting the parameter `extend` (default: 400bp). With this, each region is extended to a length of 400bp either along the + or along the - direction as specified by the dependent strand information. The `extend` value will not be added to the given length of the short reads but the final length of the extended reads will be the length as specified by the `extend` parameter.

You can create the genome vector by typing

```
> CONTROL.SET = MEDIPS.genomeVector(data = CONTROL.SET, bin_size = 50,
+   extend = 400)
```

For each pre-defined genomic bin, the genome vector stores the number of provided overlapping extended short reads and these are interpreted as the raw MeDIP-Seq signals. After having called the `MEDIPS.genomeVector` function, the slots of the MEDIPS SET associated to the genome vector are occupied. For example, there is a slot that contains the raw signals for each genomic bin.

Based on the total number of provided short reads (n), the raw MeDIP-Seq signals can be transformed into a reads per million (rpm) format in order to assure that coverage profiles derived from different biological samples are comparable, although generated from differing amounts of short reads. Let x_{bin_i} be the raw MeDIP-Seq signal of the genomic bin i , where $i=1, \dots, m$ and m is the total number of genomic bins, then the rpm value of the genomic bin is simply defined as:

$$rpm_{bin_i} = \frac{x_{bin_i} \cdot 10^6}{n}$$

It is already possible to export the raw signals in a reads per million (rpm) format as a wiggle (WIG) file by typing:

```
> MEDIPS.exportWIG(file = "output.rpm.control.WIG", data = CONTROL.SET,
+   raw = T, descr = "hESCs.rpm")
```

At this point, it is necessary to set the parameter `raw=T`. Otherwise, the export function tries to write out the normalized data that does not exist yet. The `descr` parameter contains an arbitrary description for the wiggle file and will be visualized by a suitable genome browser. It is recommended to gzip the WIG file before you upload it to e.g. the UCSC browser.

3.3 Saturation Analysis

The saturation analysis addresses the question, whether the number of input regions is sufficient to generate a saturated and reproducible methylation profile of the reference genome. The main idea is that an insufficient number of short reads will not result in a saturated methylation profile. Only if there is a

sufficient number of short reads, the resulting genome wide methylation profile will be reproducible by another independent set of a similar number of short reads.

You can start the saturation analysis by typing

```
> sr.control = MEDIPS.saturationAnalysis(data = CONTROL.SET, bin_size = 50,
+     extend = 400, no_iterations = 10, no_random_iterations = 1)
```

For the saturation analysis, the total set of available regions is divided into two distinct random sets (A and B) of equal size. In our example, both sets A and B will contain approx. 2,730,631 randomly selected regions. Both sets A and B are again divided into random subsets of equal size where the number of subsets is determined by the parameter `no_iterations` (default=10). In our example, each subset of A (A_1, A_2, \dots, A_{10}) and of `textttB` (B_1, B_2, \dots, B_{10}) will contain approx. 273,063 randomly selected reads. For each set, A and B, the saturation analysis iteratively selects an increasing number of subsets and creates according genome vectors as described in section 3.2. Here, the parameters `bin_size` and `extend` fulfill the same tasks as described in section 3.2. In case, these parameters remain un-specified, MEDIPS accesses the parameter settings as specified previously for generating the genome vector.

In each iteration step, the resulting genome vectors for the subsets of A and B are compared using pearson correlation. As the number of considered regions increases during each iteration step, it is assumed that the resulting genome vectors become more similar, a dependency that is expressed by an increased correlation.

Because such a saturation analysis can be performed on two independent sets of short reads only, a true saturation can only be calculated for half of the available short reads. As it is of interest to examine the reproducibility of the MeDIP-Seq experiment for the total set of available short reads, the saturation analysis is always followed by an estimated saturation analysis. For the estimated saturation analysis, the full set of given regions is doubled by considering each region twice and then the described saturation analysis is performed on the artificially doubled set of regions.

The saturation analysis does not modify the MEDIPS SET object but the results are stored at the specified saturation results object (here `sr.control`). The results of the saturation and of the estimated saturation analysis can be viewed by typing

```
> sr.control

$distinctSets
      [,1]      [,2]
[1,]      0 0.0000000
[2,] 273063 0.3853767
[3,] 546126 0.5528254
[4,] 819189 0.6496502
[5,] 1092252 0.7122838
[6,] 1365315 0.7558392
[7,] 1638378 0.7880560
[8,] 1911441 0.8125255
```

```
[9,] 2184504 0.8321426
[10,] 2457567 0.8478523
[11,] 2730631 0.8607522
```

```
$estimation
      [,1]      [,2]
[1,]      0 0.0000000
[2,] 273063 0.3904055
[3,] 546126 0.5639273
[4,] 819189 0.6574232
[5,] 1092252 0.7196482
[6,] 1365315 0.7630306
[7,] 1638378 0.7944670
[8,] 1911441 0.8184046
[9,] 2184504 0.8375583
[10,] 2457567 0.8531783
[11,] 2730630 0.8658112
[12,] 3003693 0.8766087
[13,] 3276756 0.8856613
[14,] 3549819 0.8935581
[15,] 3822882 0.9003433
[16,] 4095945 0.9063547
[17,] 4369008 0.9116673
[18,] 4642071 0.9164515
[19,] 4915134 0.9206184
[20,] 5188197 0.9244991
[21,] 5461263 0.9280434
```

```
$numberReads
[1] 5461263
```

```
$maxEstCor
[1] 5.461263e+06 9.280434e-01
```

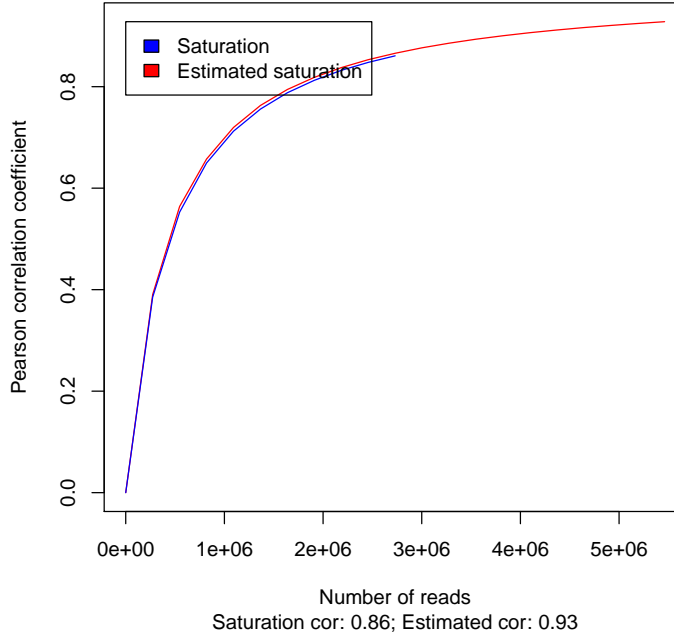
```
$maxTruCor
[1] 2.730631e+06 8.607522e-01
```

The maximal obtained correlation of the saturation analysis is stored at the `maxTruCor` slot and the maximal obtained correlation of the estimated saturation analysis is stored at the `maxEstCor` slot of the saturation results object (first column: total number of considered reads, second column: obtained correlation). The results of each iteration step are stored in the `distinctSets` and `estimation` slots for the saturation and estimated saturation analysis, respectively (first column: total number of considered reads, second column: obtained correlation).

These results can be visualized by typing

```
> MEDIPS.plotSaturation(sr.control)
```


Saturation analysis



Because the artificially doubled set of short reads, as utilized for the estimated saturation analysis, does not represent a true outcome of a MeDIP-Seq experiment, the calculated correlations will overestimate the true reproducibility. It is assumed that the true correlation for the full set of available short reads will be between the results of the true (0.86) and of the estimated (0.93) saturation analysis.

As an example, the results show that a number of approx. 1.09 million reads will result in a genome wide methylation profile that has a correlation of only 0.71 compared to a methylation profile obtained by another independent set of approx. 1.09 million reads. On the other hand, it can be deduced that two independent sets of approx. 4.3 million reads return methylation profiles having a correlation of 0.91 when compared. Even when the total set of reads is increased to e.g. 5.4 million, the resulting correlation only increases to 0.93. Therefore, it might be necessary to produce a total number of approximately 4 million reads for the chromosomes 1, 2, and 3 of the human genome. Obviously, the quality of a reconstructed methylome always increases by an increasing number of generated short reads. However, the saturation analysis assists for rating whether the costs of additional sequencing runs is in proportion to the impact on the quality of the reconstructed methylome. Please note, the results of the saturation analysis are dependent on the size of the examined reference genome (here, only the chromosomes 1, 2, and 3 are considered).

Further parameters that can be specified for the saturation analysis are:

- **no_random_iterations:** approaches that randomly select data entries may be processed several times in order to obtain more stable results. By specifying the **no_random_iterations** parameter (default=1) it is possible to run the saturation analysis several times. The final results returned

to the saturation results object are the averaged results of each random iteration step.

- **empty_bins**: can be either TRUE or FALSE (default TRUE). This parameter effects the way of calculating correlations between the resulting genome vectors. A genome vector consists of concatenated vectors for each included chromosome. The size of the vectors is defined by the **bin_size** parameter. If there occur genomic bins which contain no overlapping regions, neither from the subsets of A nor from the subsets of B, these bins will be neglected when the parameter is set to FALSE.
- **rank**: can be either TRUE or FALSE (default FALSE). This parameter also effects the way of calculating correlations between the resulting genome vectors. If **rank** is set to TRUE, the correlation will be calculated for the ranks of the bins instead of considering the counts. Setting this parameter to TRUE is a more robust approach that reduces the effect of possible occurring outliers (these are bins with a very high number of overlapping regions) to the correlation.

3.4 Receiving DNA Sequence Pattern Positions

The idea of a MeDIP experiment is to identify methylated cytosines. For this, an antibody is used that recognises methylated cytosines. However, it has been shown [Down et al., 2008], [Pelizzola et al., 2008] that MeDIP signals scale with local densities of CpGs and are not necessarily influenced by only methylated cytosines. In order to integrate the information about CpG densities into the following analysis, it is necessary to identify the genomic positions of all CpGs. This can be achieved by typing

```
> CONTROL.SET = MEDIPS.getPositions(data = CONTROL.SET, pattern = "CG")
```

MEDIPS returns all start positions of CpGs on the plus strand of the reference genome. As the CG pattern is reverse complementary, it is only necessary to scan the plus strand. The pattern dependent slots of the MEDIPS SET object now store the necessary informations. (Have a look by just typing the name of your MEDIPS SET object). In principle, it is possible to identify the positions of any other sequence pattern. For example, Lister and colleagues [Lister et al., 2009] have shown that in human embryonic stem cells, methylation occurs at cytosines outside of the CpG context. Therefore, it might be worthwhile to scan for all cytosines within the reference genome. Please note, for sequence patterns that are not reverse complementary, the function returns all start positions of the pattern within the plus and the minus strand. But for now, we will continue using the CpG pattern.

3.5 Sequence Pattern Coverage Analysis

The main idea of the coverage analysis is to test the number of CpGs (or any other predefined sequence pattern, see section 3.4) covered by the given short reads and to have a look at the depth of coverage. Before the coverage analysis can be executed, it is necessary to previously execute the `MEDIPS.getPositions` function (see section 3.4). Afterward, the coverage analysis can be started by typing

```
> cr.control = MEDIPS.coverageAnalysis(data = CONTROL.SET, extend = 400,
+   no_iterations = 10)
```

For the coverage analysis, the total set of available regions is divided into random subsets of equal size where the number of subsets is determined by the parameter `no_iterations` (default=10). The coverage analysis iteratively selects an increasing number of subsets and tests how many CpGs are covered by the available regions. Moreover, it is tested how many CpGs are covered at least 1x, 2x, 3x, 4x, 5x, and 10x. These levels of coverage depths can be adjusted by setting the `coverages` parameter (see below). As the regions are typically of short length (e.g. 36bp), it is recommended to extend the region length by an `extend` value (see section 3.2).

The coverage analysis does not modify the MEDIPS SET object but the results are stored at the specified coverage results object (here `cr.control`). The results of the coverage analysis can be viewed by typing

```
> cr.control

$matrix
      [,1] [,2] [,3] [,4] [,5] [,6] [,7]
[1,]     0     1     2     3     4     5    10
[2,]     0     0     0     0     0     0     0
[3,] 546126 2092778 845511 376014 181490 94682 6706
[4,] 1092252 3006407 1662524 968001 589163 371798 57489
[5,] 1638378 3544288 2259693 1494593 1013303 704312 158297
[6,] 2184504 3903754 2702558 1930600 1404690 1036973 287456
[7,] 2730630 4162119 3043433 2289891 1746465 1348527 435008
[8,] 3276756 4359697 3313480 2585243 2044027 1630069 592404
[9,] 3822882 4516814 3535989 2834530 2301875 1882166 751829
[10,] 4369008 4643784 3719539 3047522 2525826 2107476 913806
[11,] 4915134 4749503 3873861 3230269 2721836 2307826 1073454
[12,] 5461263 4839588 4007430 3388306 2896087 2487659 1227601

$maxPos
[1] 6072451

$pattern
[1] "CG"

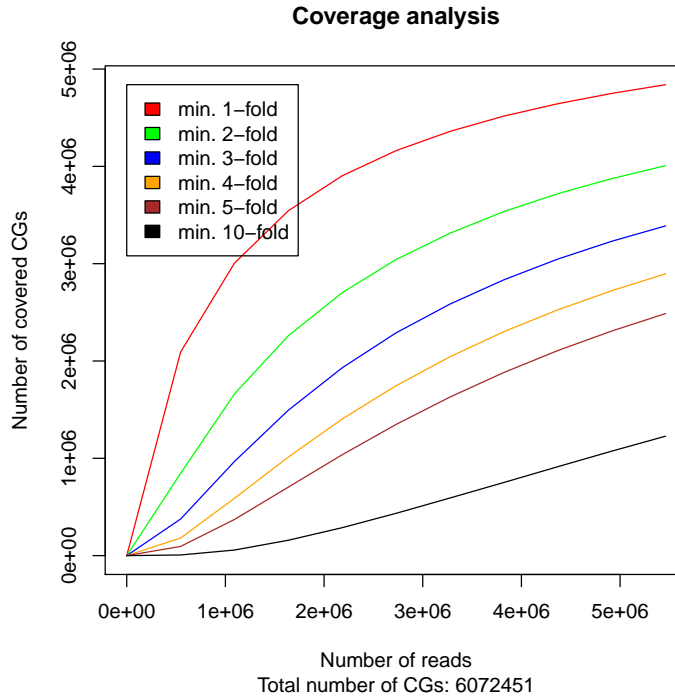
$coveredPos
      [,1] [,2] [,3] [,4] [,5] [,6]
[1,]     1.0     2.00     3.00     4.00     5.00    10.0
[2,] 4839588.0 4007430.00 3388306.00 2896087.00 2487659.00 1227601.0
[3,]     0.8     0.66     0.56     0.48     0.41     0.2
```

For example, the `maxPos` slot shows the total number of CpGs within the chromosomes 1, 2, and 3 of the reference genome. Moreover, the `matrix` slot shows the results of the coverage analysis for each iteration and for each of the tested coverage depths. Here, the first row shows the tested levels of coverage. The first column contains the number of considered short reads in each iteration.

The following columns show the number of sequence patterns covered (at least) by the according coverage depths.

The results of the coverage analysis can be visualized by typing

```
> MEDIPS.plotCoverage(cr.control)
```



The plot shows that a number of approx. 1.09 million extended short reads cover 3 million of the 6,072,451 CpGs at least one time. On the other hand, approx. 4.3 million reads cover 4.6 million of all CpGs at least one time. This number of covered CpGs can be increased to approx. 4.8 million, by increasing the total number of short reads to 5.4 million.

Further parameters that can be specified for the coverage analysis are:

- **no_random_iterations**: approaches that randomly select data entries may be processed several times in order to obtain more stable results. By specifying the **no_random_iterations** parameter (default=1) it is possible to run the coverage analysis several times. The final results returned to the coverage results object are the averaged results of each random iteration step.
- **coverages**: default is `c(1, 2, 3, 4, 5, 10)`. The coverages define the depth levels for testing how often a sequence pattern was covered by the given regions. Just specify any other vector of coverage depths you would like to test.

Although an increase of short reads will always improve the sequence pattern coverage, the coverage analysis, together with the saturation analysis, allow

for gaining an impression on the overall sequence pattern coverage and reproducibility of reconstructing a methylome based on the total number of MeDIP-Seq derived short reads. These data quality controls assist in deciding whether the costs of additional experimental runs are in due proportion to the expected improvements on coverage and reproducibility.

3.6 CpG Enrichment

As a quality check for the enrichment of CpG rich DNA fragments obtained by the immunoprecipitation step of a MeDIP experiment, MEDIPS provides the functionality to calculate CpG enrichment values. The main idea is to check, how strong the regions are enriched for CpGs compared to the reference genome. For this, MEDIPS counts the number of Cs, the number ofGs, the number CpGs, and the total number of bases within the specified reference genome. Subsequently, MEDIPS calculates the relative frequency of CpGs and the observed/expected ratio of CpGs present in the reference genome. Additionally, MEDIPS calculates the same for the DNA sequences underlying the given regions. The final enrichment values result by dividing the relative frequency of CpGs (or the observed/expected value, respectively) of the regions by the relative frequency of CpGs (or the observed/expected value, respectively) of the reference genome. (See also Supplementary Material in [Chavez et al., 2010].)

You can start the CpG enrichment analysis by typing

```
> er.control = MEDIPS.CpGenrich(data = CONTROL.SET)
```

The CpG enrichment analysis does not modify the MEDIPS SET object but the results are stored at the specified enrichment results object (here `er.control`). You can access the results by typing

```
> er.control
```

```
$regions.CG  
[1] 3796311
```

```
$regions.C  
[1] 47803260
```

```
$regions.G  
[1] 47926519
```

```
$regions.relH  
[1] 1.878741
```

```
$regions.GoGe  
[1] 0.3348291
```

```
$genome.C  
[1] 592445731
```

```
$genome.G  
[1] 592804204
```

```

$genome.CG
[1] 28670425

$genome.relH
[1] 0.9903856

$genome.GoGe
[1] 0.2363220

$enrichment.score.relH
[1] 1.896980

$enrichment.score.GoGe
[1] 1.416834

```

The enrichment results object contains several slots that show the number of Cs, Gs, and CpGs within the reference genome and within the given regions. Additionally, there are slots that show the relative frequency as well as the observed/expected CpG ratio within the reference genome and within the given regions. Finally, the slots `enrichment.score.relH` and `enrichment.score.GoGe` indicate the enrichment of CpGs within the given regions compared to the reference genome. For short reads derived from an INPUT experiment (that is sequencing of none-enriched DNA fragments), the enrichment values should be close to 1 (see an example in section 3.11). In contrast, a MeDIP-Seq experiment should return CpG rich sequences what will be indicated by increased enrichment scores. In our example, the enrichment score for the relative CpG enrichment is 1.896980, indicating an enrichment of CpG rich regions.

In case you would like to examine not only the regions defined by the short reads, but also the DNA sequences of the putative longer DNA fragments from where the short reads were derived, it is possible to increase the length of the regions by specifying the `extend` (default NULL) parameter. By setting the `extend` to any positive value, the regions will be extended to the plus or to the minus direction (dependent on the strand information of the reads) and afterwards the CpG enrichment will be calculated for the extended regions.

3.7 Creating the Coupling Vector

The need for MeDIP-Seq data correction occurs through an unspecific binding of the utilized antibody to unmethylated cytosines, especially in the context of low densities of methylated cytosines. Similar to other MeDIP normalization methods [Down et al., 2008], [Pelizzola et al., 2008], MEDIPS tries to correct for this effect by incorporating local CpG densities into the MeDIP-Seq signals. In order to correct for local CpG densities, it is necessary to calculate a coupling vector first. The coupling vector is of the same size as the predefined genome vector (see also section 3.2) but contains local CpG densities (also called coupling factors) instead of the raw signals for each genomic bin. Before the coupling vector can be created, it is necessary to previously execute the `MEDIPS.getPositions` function (see section 3.4).

The coupling vector is created and attached to the MEDIPS SET object by typing e.g.

```
> CONTROL.SET = MEDIPS.couplingVector(data = CONTROL.SET, fragmentLength = 700,  
+   func = "count")
```

For each pre-defined genomic bin, the density of surrounding CpGs (or of another pre-defined sequence pattern, respectively, see section 3.4) is calculated. For this, first a maximal distance has to be defined by specifying the parameter `fragmentLength`. Only CpGs within the range of `[(bin_position-fragmentLength), bin_position+fragmentLength]` will contribute to the final local coupling factor. The optimized value for the `fragmentLength` parameter will reflect the estimated size of your sonicated DNA fragments. There are several ways for calculating a coupling factor for a genomic bin. The simplest way is to count the number of CpGs within the maximal defined distance around a genomic bin. Another approach is to weight each CpG by its distance to the current genomic bin. CpGs further away from the current genomic bin will receive smaller weights, whereas CpGs close to the genomic bin will receive higher weights. Again, there are several possible ways for such a weighting function. MEDIPS supports setting the weighting function parameter `func` to:

- **count**: simply count the number of CpGs within the predefined maximal distance to the current bin
- **linear**: the weights for CpGs decreases in a linear way and end at 0 at the predefined maximal distance to the current bin
- **exp**: the weights for CpGs decreases in an exponential way [Pelizzola et al., 2008]
- **log**: the weights for CpGs decreases in a logarithmic way [Pelizzola et al., 2008]
- **custom**: by setting the parameter to custom, it is required to specify a custom distance weights file using the parameter `distFile`. For example, [Down et al., 2008] have generated distance weights in an empirical way. From their results of the empirical approach, we have created the file `flat_400_700.tab` that can be downloaded from <http://medips.molgen.mpg.de>. You can create any of such a distance file by your own and specify it here. Here, the `fragmentLength` parameter will be neglected and the maximal distance within your provided distance file will be the limit.

We have systematically calculated coupling factors with varying `fragmentLength` and `func` parameters and compared the resulting coupling vectors to DNA-methylation values derived from bisulphite experiments performed by the human epigenome project (HEP) [Eckhardt et al., 2006]. The best negative correlation (that is the higher the CpG density, the lower the bisulfite derived methylation values) was achieved by setting the parameters to `fragmentLength=700` and `func=count` (see Supplementary Material in [Chavez et al., 2010]).

The coupling vector can be exported into a wiggle file by typing

```
> MEDIPS.exportWIG(file = "PatternDensity.WIG", data = CONTROL.SET,
+   pattern.density = TRUE, descr = "Pattern.density")
```

The exported Wiggle file can be uploaded into common genome browsers and allows for visualizing the density of the specified sequence pattern (e.g. CpGs) along the chromosomes. We recommend to gzip the file before uploading it to the genome browser.

3.8 Calibration Curve and Linear Regression

As we have created a genome vector containing the raw signals at each genomic bin as well as an according coupling vector containing coupling factors at each genomic bin (both stored within the MEDIPS SET object), we can now examine the dependency of local MeDIP-Seq signal intensities and local CpG densities. This dependency can be made tangible by calculating the calibration curve:

```
> CONTROL.SET = MEDIPS.calibrationCurve(data = CONTROL.SET)
```

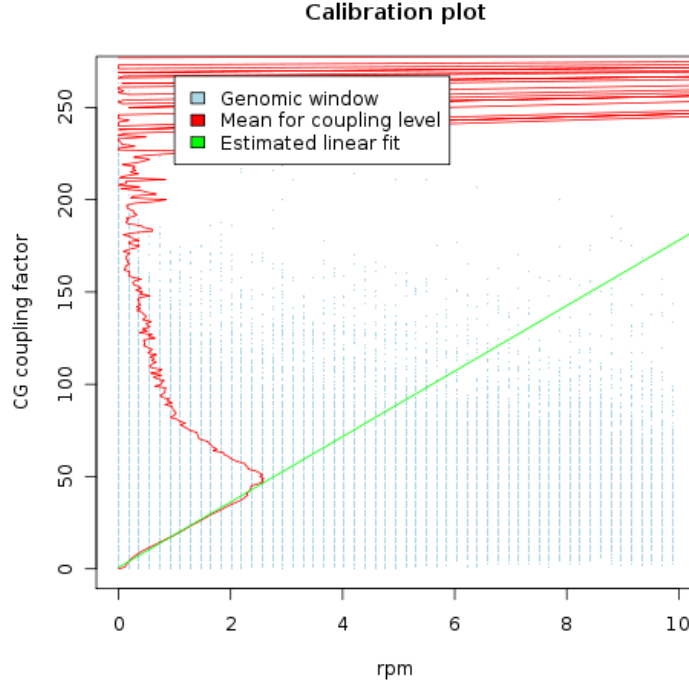
Calculation of the calibration curve is achieved by first dividing the total range of coupling factors into regular levels. Second, all genomic bins are partitioned into these levels by considering their associated coupling factors. Finally, for each level of coupling factors, MEDIPS calculates the mean raw signal and mean coupling factor of all genomic bins that fall into this level. (For a detailed description see Supplementary Material in [Chavez et al., 2010].)

The calibration curve represents averaged signals and coupling factors over the full range of coupling factors. It indicates the experiment specific dependency between local signal intensities and CpG densities. The results of the calibration curve calculation can be visualized by typing e.g.

```
> MEDIPS.plotCalibrationPlot(data = CONTROL.SET, linearFit = T,
+   plot_chr = "chr1", rpm = T, xrange = 10)
```

Because the amount of data to be plotted can become very huge when plotting full genome data, it is strongly recommended to call a e.g. `png("plot.png")` function before calling this plot command. The plot will be available in the specified file (do not forget the `dev.off()` command after the plotting command). Otherwise, R might not be able to visualize the plot in reasonable time. Each data value within the calibration plot represents a genomic bin. The x-axis shows the raw signals and the y-axis shows the coupling factors for the genomic bins. The red curve represents the calibration curve.

The calibration curve reveals that, in average, an increase of MeDIP-Seq signals is caused by an increasing CpG density. This approximately linear dependency is visible for the low range of coupling factors, only. For higher levels of CpG densities, the mean MeDIP-Seq signals decrease. It is assumed that this decrease is caused by the fact that in mammalian cells, regions of higher CpG densities are mainly unmethylated. In agreement with this assumption, Pelizzola and colleagues [Pelizzola et al., 2008] have shown that the dependency of MeDIP derived signals and CpG density continues for higher levels of CpG densities, by analyzing artificially fully methylated samples using MeDIP-Chip. In detail, they have identified a sigmoidal dependency between CpG density and MeDIP-Chip data. In agreement with Pelizzola et al. [Pelizzola et al.,



2008], it is assumed that their signal plateau in the lower range of chip signals is caused by background noise, but in contrast to Pelizzola et al., it is assumed that their signal plateau in the upper range of chip signals occurs by a saturation of hybridization events and is therefore an array specific artefact. Motivated by the observations made by Pelizzola et al. [Pelizzola et al., 2008] and by visual inspection of the MeDIP-Seq derived calibration curve, a continuing linear dependency of MeDIP-Seq signals for higher levels of CpG densities is assumed. Analogous to Down et al. [Down et al., 2008], the local maximum of mean MeDIP-Seq signals of the calibration curve in the lower part of coupling factors is identified. Let

$$y = y_1, \dots, y_l$$

be the mean coupling factors, and let

$$x = x_1, \dots, x_l$$

be the according mean MeDIP-Seq signals of the calibration curve, where l is the number of tested coupling factor levels and $i = 1, \dots, l$, then the smallest level i is identified, where

$$x_{i-3} \leq x_{i-2} \leq x_{i-1} \leq x_i \geq x_{i+1} \geq x_{i+2} \geq x_{i+3}$$

Let i_{max} be the according identified level of i , then

$$y_{max} = y_1, \dots, y_{i_{max}}$$

$$x_{max} = x_1, \dots, x_{i_{max}}$$

are the parts of the calibration curve in the low range of coupling factors, where an approximately linear dependency between MeDIP-Seq signals and coupling factors is observed. Here, x_{max} can be explained by a function of y_{max} as

$$x_{max} = f(y_{max}) + \epsilon$$

where ϵ is an error variable (i.e. measurement errors) that is expected to spread by chance and therefore, its expectation value is $E(\epsilon) = 0$. Because a linear dependency between x_{max} and y_{max} is assumed, x_{max} can be described as

$$x_{max} = \alpha + \beta \cdot y_{max} + \epsilon$$

where the parameter α is the theoretical y-intercept, and the parameter β is the theoretical slope. Based on the pre-calculated x_{max} and y_{max} vectors, linear regression is performed, in order to identify a suitable linear model. Linear regression estimates regression coefficients a and b for the parameters α and β so that it is valid:

$$x_{max_i} = a + b \cdot y_{max_i} + e_i$$

where $i = 1, \dots, i_{max}$. Here, the residuum e_i reflects the difference between the regression curve $a + b \cdot y_{max_i}$ and the measurements of x_{max_i} . Moreover, x_{max_i} can be replaced by an estimate \hat{x}_{max_i} , where

$$x_{max_i} - \hat{x}_{max_i} = e_i$$

and therefore, it is valid:

$$\hat{x}_{max_i} = a + b \cdot y_{max_i}$$

After having calculated the calibration curve, the `MEDIPS.calibrationCurve()` function performs the described linear regression and stores concrete values for the parameters a (intercept) and b (slope) within the according slots of the MEDIPS SET. By accessing the received parameters a and b , concrete values for the parameter \hat{x}_{max_i} can be calculated by the latter formula. For the low range of coupling factors, these estimates model the observed progression of the calibration curve. As discussed above, a continuing linear dependency between MeDIP-Seq signals and CpG density is expected for the higher range of coupling factors. Based on the obtained linear model parameters, concrete \hat{x}_{max_i} values can be calculated for the full range of coupling factors. Therefore,

$$\hat{x} = \hat{x}_1, \dots, \hat{x}_{max_i}, \dots, \hat{x}_l$$

are the estimated mean MeDIP-Seq signals over the full range of coupling factor levels l , calculated with respect to the obtained linear model parameters.

When the parameter `linearFit` of the `MEDIPS.plotCalibrationPlot()` function was set to `TRUE`, the calibration plot contains a linear curve (green curve) that visualizes the results of the performed linear regression. This curve represents the calculated linear dependency between signals and CpG densities as estimated from the low range of coupling factors.

Further parameters that can be specified when plotting the calibration curve are:

- **plot_chr**: default="all". Please don't forget to call a e.g. `png("file.png")` function before calling the plot command using `all` (see above). Alternatively, you can specify a selected chromosome (e.g. `chr1`). Here, the `plot_chr` parameter only affects the plot and does not affect the MEDIPS SET object.
- **xrange**: The mean signal range of the calibration curve typically falls into a low signal range. By setting the `xrange` parameter to e.g. 50 (suitable for raw data), the calibration plot will only plot genomic bins associated with signals ≤ 50 . Therefore, the effect of an increased CpG density to an increased signal can be better visualized, especially if the data contains genomic bins with high signals.
- **rpm**: can be either TRUE or FALSE. If set to TRUE, the signals will be transformed into reads per million (rpm) before plotted. Additionally, the mean signal values of the calibration curve and of the estimated linear curve will be transformed to rpm scale. The coupling values remain untouched.

The calibration plot is very characteristic for MeDIP-Seq experiments. The quality of the enrichment step of the MeDIP experiment can be estimated by visual inspection of the progression of the calibration curve. Calibration curves for data derived from INPUT experiments look different (please see an example in section 3.11).

3.9 Relative Methylation Score and Export of Normalized Data

As soon as the normalization parameters are calculated (see previous section), the raw signals will be normalized and the normalized data will be stored within the MEDIPS SET object.

```
> CONTROL.SET = MEDIPS.normalize(data = CONTROL.SET)
```

For MeDIP-Seq data normalization, \hat{x} (see section 3.8) is utilized in order to weight the observed MeDIP-Seq signals of the genomic bins with respect to their associated coupling factors. Let (x_{bin_i}, y_{bin_i}) be the raw MeDIP-Seq signal of the genomic bin i (i.e. the number of overlapping extended short reads), and the pre-calculated coupling factor at the genomic bin i , where $i = 1, \dots, m$ and m is the total number of genomic bins, then the normalized relative methylation score is defined as

$$rms_{bin_i} = \frac{x_{bin_i} \cdot 10^6}{(a + b \cdot y_{bin_i}) \cdot n} = \frac{x_{bin_i} \cdot 10^6}{\hat{x}_{bin_i} \cdot n}$$

where $\hat{x}_{bin_i} = a + b \cdot y_{bin_i}$ is the estimated weighting parameter obtained by considering the coupling factor y_{bin_i} of the genomic bin i , and a and b are the pre-calculated regression parameters. Based on the total number of short reads (n), the raw MeDIP-Seq signals are, in parallel, transformed into a reads per million format in order to assure that rms values are comparable between methylomes generated from differing amounts of short reads.

The rms values can be exported into a wiggle file by typing

```
> MEDIPS.exportWIG(file = "output.rms.control.WIG", data = CONTROL.SET,
+   raw = F, descr = "hESCs.rms")
```

All slots of the MEDIPS SET are now occupied. You can again have a look at the MEDIPS SET by typing

```
> CONTROL.SET
```

```
S4 Object of class MEDIPSset
=====
Regions informations
=====
Regions file:  MeDIP_hESCs_chr1-3.txt
Organism:  BSgenome.Hsapiens.UCSC.hg19
Chromosomes: chr1 chr2 chr3
Chromosome lengths: 249250621 243199373 198022430
Number of regions: 5461263
Regions chromosomes: chr1 chr1 chr1...
Regions start positions: 100001075 10000155 10000178...
Regions stop positions: 100001111 10000191 10000214...
Regions strand: - - -...
=====
Genome vector signals informations
=====
Genome wide bin size: 50
Reads extended by: 400
Genome vector chromosomes: chr1 chr1 chr1...
Genome vector positions: 1 51 101...
Genome vector signals: 0 0 0...
=====
Pattern informations
=====
Pattern: CG
Number of patterns: 6072451
Pattern chromosomes: chr1 chr1 chr1...
Pattern positions: 10469 10471 10484...
=====
Genome vector coupling factor informations
=====
Distance function: count
Distance file: empty
Fragment length: 700
Genome vector coupling factors: 0 0 0...
=====
Calibration informations
=====
Calibration curve mean signals: 0.01871272 0.6301572 0.7357637...
Calibration curve mean coupling factors: 0 1 2...
Calibration curve variance: NA NA NA...
Intercept: 0.737575375951728
Slope: 3.24687748777487
```

```

Calibration chromosome:  all
=====
Genome vector normalized signal informations
=====
Normalization output interval: [0:1000]
Genome vector normalized signals: 0 0 0...

```

3.10 Methylation Profiles and Absolute Methylation Score

A typical question of MeDIP based DNA-methylation experiments is to examine the methylation state of specific genomic regions, like e.g. CpG islands, promoters and other regions of interest (ROI).

MEDIPS provides the functionalities to calculate averaged methylation values for either pre-defined ROIs or for genome wide windows. In order to receive the methylation state of targeted genomic regions, first a ROI file has to be created. The required structure of any file containing regions of interest is:

- the first column contains the chromosome of the ROI (e.g. chr1)
- the second column contains the start position of the ROI
- the third column contains the stop position of the ROI
- the fourth column contains an identifier of the ROI

As an example, you can download the file `hg19.chr1-3.txt` from <http://medips.molgen.mpg.de>.

The file contains hg19 promoter regions of Ensembl transcripts located on the chromosomes 1, 2, and 3 as received from www.biomart.org. Here, the genomic coordinates start at -1kb of the transcript start sites (TSSs) and stop at +0.5kb downstream of the TSSs. Mean methylation values for these regions can be already summarized for the generated MEDIPS SET object by typing:

```

> promoter = MEDIPS.methylProfiling(data1 = CONTROL.SET, ROI_file = "hg19.chr1-3.txt",
+   math = mean, select = 2)

```

Further parameters that can be specified are:

- **chr**: only the specified chromosome will be evaluated (e.g. chr1)
- **select**: can be either 1 or 2. If set to 1, variances will be calculated based on the rpm values; if set to 2, variances will be calculated based on the rms values.
- **math**: default=`mean`; Here, you can specify other functions available in R for summarizing values like `median` or `sum`.
- **data2**: default=`NULL`; Here, a second MEDIPS SET can be provided, in case two MEDIPS SETs have to be compared (see also section 3.12)
- **input**: default=`NULL`; Here, an INPUT SET can be provided. An INPUT SET is a MEDIPS SET but generated from data derived from an Input sample. An Input sample is generated by sonication of the DNA

but without a subsequent immunoprecipitation step. Therefore, Input data reflects the genomic background. MEDIPS allows for accessing Input data (if available) in order to calculate a genomic background signal distribution. Such genomic background signals can be utilized for identifying hyper-methylated regions when two MEDIPS SETs are compared (see also section 3.12).

- **frame_size**: default=NULL; In spite of calculating averaged methylation values for given regions of interest, MEDIPS allows for calculating methylation levels for genome wide windows. The **frame_size** parameter defines the size of the genomic windows to be tested. Therefore, it is obvious that either the **frame_size** or the **ROI_file** parameter has to be specified.
- **step**: default=NULL; In case the **frame_size** parameter is specified, the **step** parameter can be specified to any arbitrary positive value. The **step** parameter defines the number of bases, by which the genomic windows are shifted along the chromosomes. If **step** remains *NULL*, non overlapping adjacent genomic windows will be examined. By setting the **step** parameter to e.g. 250 bp and by setting the **frame_size** parameter to e.g. 500 bp, overlapping genomic windows will be examined, where the overlap of neighbouring genomic windows is 250 bp (see also the example below).

The results are stored as a list at the specified object (here **promoter**). All list objects are vectors of the same length, where the length is defined by the number of tested ROIs. Please note, as we have so far generated and specified only one MEDIPS SET object, the results object only contains the results like the mean rpm and rms values for one MEDIPS SET (provided at **data1**). The remaining vectors are empty and will be occupied when two MEDIPS SET objects will be compared (see section 3.12). Each row refers to a ROI, the row names contain the IDs of the provided ROIs, and the vectors of the list are:

1. **chr**: the chromosome of the ROI
2. **start**: the start position of the ROI
3. **stop**: the stop position of the ROI
4. **length**: the number of genomic bins included in the ROI
5. **coupling**: the mean coupling factor of the ROI
6. **input**: the mean rpm value of the INPUT MEDIPS SET at **input** (if a second MEDIPS SET is provided)
7. **rpm_A**: the mean rpm value for the MEDIPS SET at **data1**
8. **rpm_B**: the mean rpm value for the MEDIPS SET on **data2** (if a second MEDIPS SET is provided)
9. **rms_A**: the transformed (see below) mean rms value for the MEDIPS SET at **data1**
10. **rms_B**: the transformed (see below) mean rms value for the MEDIPS SET at **data2** (if a second MEDIPS SET is provided)

11. **ams_A**: the transformed mean absolute methylation score (see below) for the MEDIPS SET at **data1**
12. **ams_B**: the transformed mean absolute methylation score (see below) for the MEDIPS SET at **data2** (if a second MEDIPS SET is provided)
13. **var_A**: the variance of the rpm or rms values (please see the parameter **select**) of the MEDIPS SET at **data1**
14. **var_B**: the variance of the rpm or rms values (please see the parameter **select**) of the MEDIPS SET at **data2** (if a second MEDIPS SET is provided)
15. **var_co_A**: the coefficient of variance of the rpm or rms values (please see the parameter **select**) of the MEDIPS SET at **data1**
16. **var_co_B**: the coefficient of variance of the rpm or rms values (please see the parameter **select**) of the MEDIPS SET at **data2** (if a second MEDIPS SET is provided)
17. **ratio**: rpm_A/rpm_B or rms_A/rms_B, respectively (please see the parameter **select**) (if a second MEDIPS SET is provided)
18. **pvalue.wilcox**: the p.value returned by R's **wilcox.test** function for comparing the rpm values (or rms values, respectively; please see the parameter **select**) of the MEDIPS SET on **data1** and of the MEDIPS SET at **data2** (if a second MEDIPS SET is provided)
19. **pvalue.ttest**: the p.value returned by R's **t.test** function for comparing the rpm values (or rms values, respectively; please see the parameter **select**) of the MEDIPS SET on **data1** and of the MEDIPS SET at **data2** (if a second MEDIPS SET is provided)

The returned mean **rms_A** (or mean **rms_B**, respectively) values are calculated based on the pre-calculated *rms* values of the according MEDIPS SET (see section 3.9). Here, at first, the mean of all pre-calculated *rms_{bin_i}* values of those genomic bins that fall into the ROI is calculated. Next, MEDIPS transforms these mean rms values into log2 scale and subsequently transforms the resulting data range into the consistent interval [0,1000] before finally stored. Therefore, the minimal transformed mean **rms_A** (or mean **rms_B**, respectively) value will be 0 and the maximal transformed mean **rms_A** (or mean **rms_B**, respectively) will be 1000. This transformation assures that the resulting mean rms values of the tested ROIs will spread over a consistent interval. Therefore, highly methylated ROIs may be subsequently identified by selecting for those associated to e.g. **rms_A>600**. The other way round, lowly methylated ROIs may be subsequently identified by selecting for those associated to e.g. **rms_A<400**. However, methylation levels received by MeDIP-Seq experiments are basically relative. Therefore, it has to be kept in mind that the received data range ([0,1000]) reflects the relative methylation levels of the ROIs of the tested sample.

We consider the *rms* values as the normalized MeDIP-Seq signals corrected for the experiment specific effect of unspecific antibody binding. In order to

identify an absolute methylation estimate for any specified region of interest, i.e. either for any functional genomic regions like promoters or CpG islands or for genome wide windows of arbitrary length, the raw MeDIP-Seq values can be normalized into absolute methylation scores (*ams*). The absolute methylation scores additionally correct for the relative CpG density of the region of interest and therefore, allow for comparing methylation profiles of genomic regions having different CpG densities. This is especially needful, when local methylation levels are associated to further functional and regulatory mechanism like e.g. gene expression alterations. As an example, it is supposed that methylation levels of proximal promoters influence the transcription rate of the according genes. However, promoters are known to show a wide spread spectrum of CpG densities. Therefore, a fully methylated high CpG density promoter will show much higher MeDIP signals than a fully methylated low CpG density promoter, although in both cases the promoter methylation level influences the transcription rate in a comparable way. Therefore, it remains inaccurate to conclude an absolute measure of promoter methylation by comparing MeDIP-Seq derived *rpm* or *rms* signals from promoters having different CpG densities.

Let

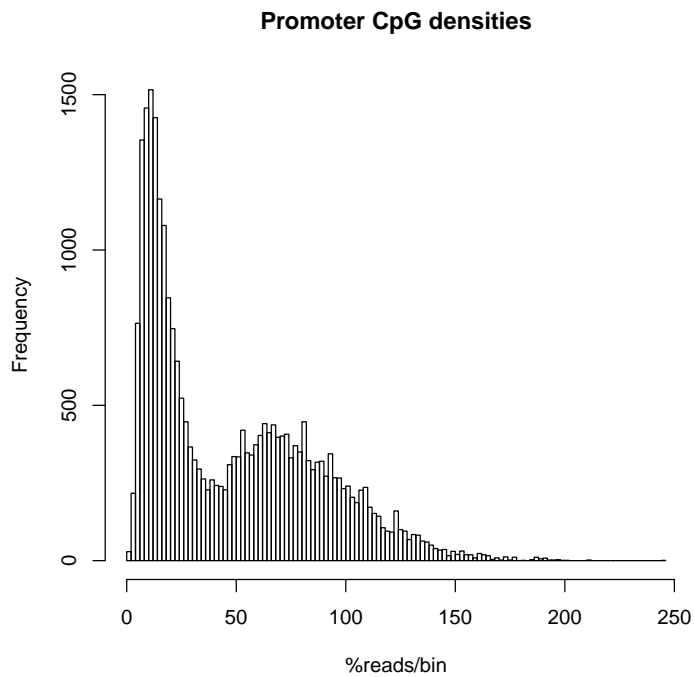
$$ROI = ((x_{bin_1}, y_{bin_1}), \dots, (x_{bin_s}, y_{bin_s}))$$

be the raw MeDIP-Seq signals and coupling factors of adjacent genomic bins i that define a region of interest (ROI), where $i = 1, \dots, s$ and s is the total number of genomic bins comprised by the ROI, then the absolute methylation score for the ROI is defined as:

$$ams_{ROI} = \frac{\frac{1}{s} \sum_{i=1}^s \frac{x_{bin_i} \cdot 10^6}{(a+b \cdot y_{bin_i}) \cdot n}}{\frac{1}{s} \sum_{i=1}^s y_{bin_i}}$$

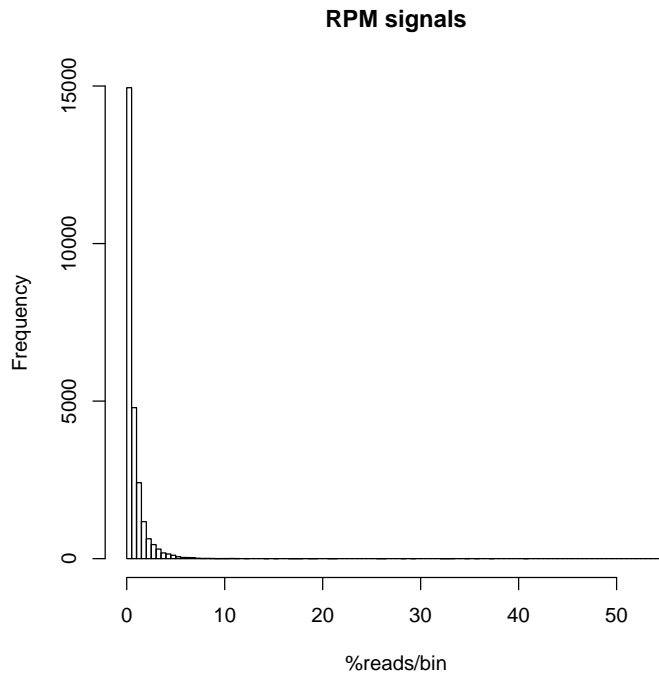
After having executed the *MEDIPS.methylProfiling()* function for the promoter regions of interest file, one can have a look at e.g. the histogram of CpG densities or methylation levels for the total set of ROIs. Simply call *Rs.hist()* function and specify the desired column of the matrix. For the mean coupling factors type e.g.

```
> hist(promoter$coupling, breaks = 100, main = "Promoter CpG densities",
+       xlab = "%reads/bin")
```

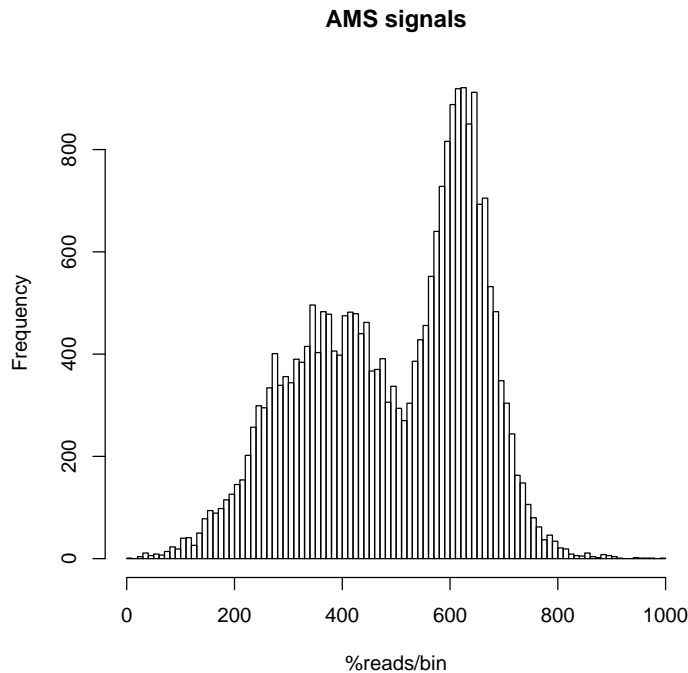
The histogram shows a bimodal distribution of promoter CpG densities. For a histogram of mean *rpm* values type

```
> hist(promoter$rpm_A[promoter$rpm_A != 0], breaks = 100, main = "RPM signals",
+       xlab = "%reads/bin")
```



Because there are promoter regions without any signals, we do not consider them for the histogram. The plot shows that a bimodal methylation distribution of the promoters is not visible for the *rpm* signals. Only the *rms* or even stronger, the *ams* values reveal the bimodal distribution of promoter methylation:

```
> hist(promoter$ams_A[promoter$ams_A != 0], breaks = 100, main = "AMS signals",
+       xlab = "%reads/bin")
```



We have shown [Chavez et al., 2010] that such summarized methylation values for ROIs, especially the *ams* values are most suitable for comparing MeDIP data to e.g. bisulphite sequencing or whole genome shotgun bisulphite sequencing data.

Besides summarizing methylation values for pre-defined ROIs, MEDIPS allows for calculating mean methylation values along the full chromosomes. For this, you have to specify a desired frame size using the parameter `frame_size`. Additionally, you can specify the `step` parameter. The `step` parameter defines the number of bases by which the frames are shifted along the chromosome. If you e.g. set the `frame_size` parameter to 500 and the `step` parameter to 250, then MEDIPS calculates mean methylation values for overlapping 500bp windows, where the size of the overlap will be 250bp for all neighbouring windows. Without specifying the `step` parameter, MEDIPS will calculate mean methylation values for all none-overlapping windows of size `frame_size`.

```
> frames.frame500.step250 = MEDIPS.methylProfiling(data1 = CONTROL.SET,
+   frame_size = 500, step = 250, math = mean, select = 2)
```

Please note, the `MEDIPS.methylProfiling()` function takes a comparable long processing time when called for genome wide short windows. For example, the processing of the full human genome using overlapping 500bp windows takes approx. 10h on our hardware. Therefore, you may want to store the received matrix afterwards by using R's `write.table()` function like:

```
> write.table(frames.frame500.step250, file = "frames.chr1-3.meth.txt",
+   sep = "\t", quote = F, col.names = T, row.names = F)
```

Here, you do not need to store the `row.names` as genome wide frames will not have identifiers. When saving results received by testing ROIs provided within a ROI file, you might want to set `row.names=T` in order to keep the identifiers.

You can upload the results table at any later time into R by typing

```
> frames.frame500.step250 = read.table(file = "frames.chr1-3.meth.txt",
+   header = T)
```

3.11 Additonal MEDIPS SET Objects

Often, it is not only of interest to generate a normalized methylation profile of a given sample, but to compare and identify differentially methylated regions (DMRs) between different samples or conditions, respectively. In order to compare two different conditions, first you have to create and process a second MEDIPS SET. In our example, the file `MeDIP_DE_chr1-3.txt` contains MeDIP-Seq data of the chromosomes 1, 2, and 3 derived from human embryonic stem cells after differentiation along the endodermal lineage into definitive endoderm (DE) ([Chavez et al., 2010]). We now process the data using the same parameter settings as for the previously created `CONTROL.SET`:

```
> TREAT.SET = MEDIPS.readAlignedSequences(BSgenome = "BSgenome.Hsapiens.UCSC.hg19",
+   file = "MeDIP_DE_chr1-3.txt", numrows = 6709336)

> TREAT.SET = MEDIPS.genomeVector(data = TREAT.SET, bin_size = 50,
+   extend = 400)

> TREAT.SET = MEDIPS.getPositions(data = TREAT.SET, pattern = "CG")

> TREAT.SET = MEDIPS.couplingVector(data = TREAT.SET, fragmentLength = 700,
+   func = "count")

> TREAT.SET = MEDIPS.calibrationCurve(data = TREAT.SET)

> TREAT.SET = MEDIPS.normalize(data = TREAT.SET)
```

So far, we have the second MEDIPS SET sufficiently processed for the subsequent comparison of two MEDIPS SETs. Nevertheless, here we perform the quality controls for the `TREAT.SET` as well:

```
> sr.treat = MEDIPS.saturationAnalysis(data = TREAT.SET, bin_size = 50,
+   extend = 400, no_iterations = 10, no_random_iterations = 1)

> sr.treat
```

```
$distinctSets
      [,1]      [,2]
[1,]      0 0.0000000
[2,] 335466 0.4945991
[3,] 670932 0.6605656
[4,] 1006398 0.7447533
[5,] 1341864 0.7948635
[6,] 1677330 0.8279856
```

```

[7,] 2012796 0.8527597
[8,] 2348262 0.8711627
[9,] 2683728 0.8853959
[10,] 3019194 0.8968643
[11,] 3354668 0.9061605

```

```

$estimation
      [,1]      [,2]
[1,]      0 0.0000000
[2,] 335466 0.4958945
[3,] 670932 0.6643025
[4,] 1006398 0.7483451
[5,] 1341864 0.7984963
[6,] 1677330 0.8325006
[7,] 2012796 0.8564147
[8,] 2348262 0.8739851
[9,] 2683728 0.8880839
[10,] 3019194 0.8992293
[11,] 3354660 0.9083642
[12,] 3690126 0.9161254
[13,] 4025592 0.9225633
[14,] 4361058 0.9280288
[15,] 4696524 0.9327958
[16,] 5031990 0.9370707
[17,] 5367456 0.9407171
[18,] 5702922 0.9439893
[19,] 6038388 0.9468882
[20,] 6373854 0.9495203
[21,] 6709336 0.9519604

```

```

$numberReads
[1] 6709336

```

```

$maxEstCor
[1] 6.709336e+06 9.519604e-01

```

```

$maxTruCor
[1] 3.354668e+06 9.061605e-01

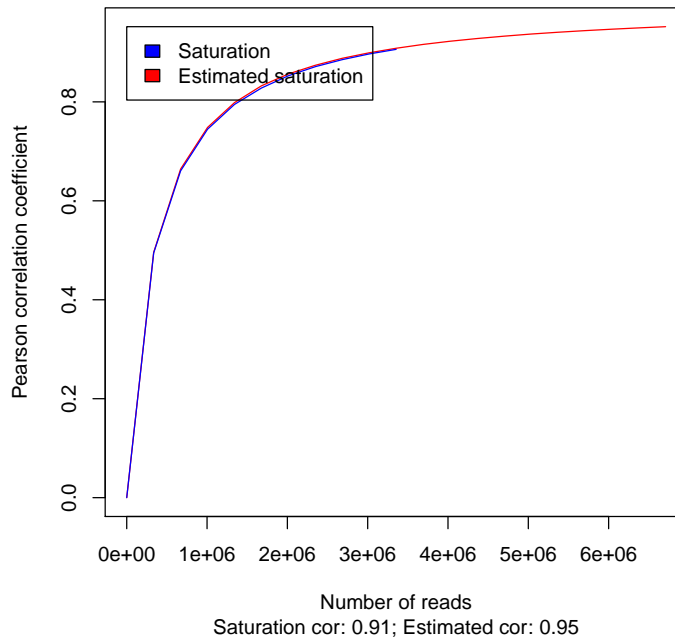
```

```

> MEDIPS.plotSaturation(sr.treat)

```

Saturation analysis



```
> cr.treat = MEDIPS.coverageAnalysis(data = TREAT.SET, extend = 400,
+   no_iterations = 10)
```

```
> cr.treat
```

```
$matrix
```

	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]	[,7]
[1,]	0	1	2	3	4	5	10
[2,]	0	0	0	0	0	0	0
[3,]	670933	2461644	1189258	635045	366405	224842	33475
[4,]	1341866	3352216	2081070	1368467	936778	662978	173165
[5,]	2012799	3847830	2669197	1938698	1443325	1097202	358529
[6,]	2683732	4174733	3084146	2372604	1864662	1485928	565013
[7,]	3354665	4409251	3397143	2713179	2207741	1818765	778559
[8,]	4025598	4587194	3640039	2985342	2494146	2102033	988961
[9,]	4696531	4729451	3837309	3209671	2732457	2347470	1190977
[10,]	5367464	4845100	4001080	3397902	2935103	2559891	1382614
[11,]	6038397	4941168	4140197	3558648	3110168	2742893	1560496
[12,]	6709336	5023127	4259222	3698722	3261285	2902361	1726187

```
$maxPos
```

```
[1] 6072451
```

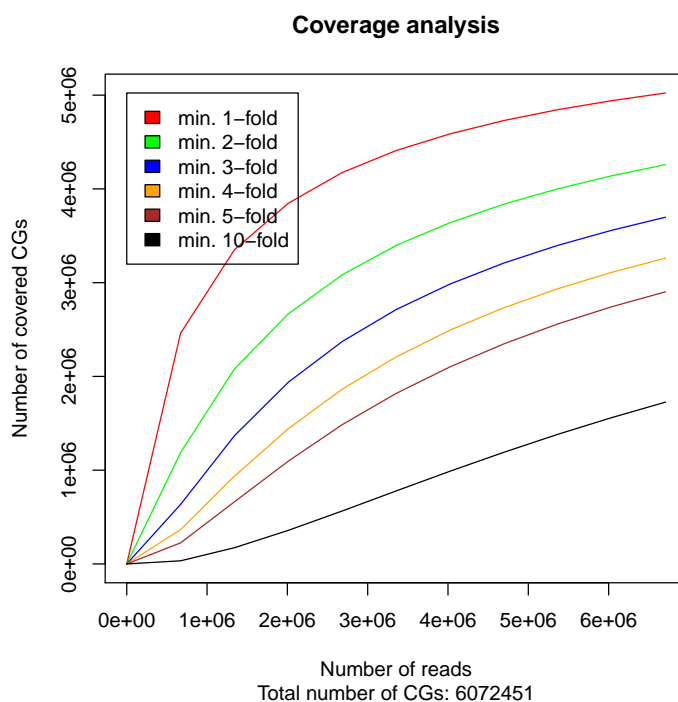
```
$pattern
```

```
[1] "CG"
```

```
$coveredPos
```

	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]
[1,]	1.00	2.0	3.00	4.00	5.00	10.00
[2,]	5023127.00	4259222.0	3698722.00	3261285.00	2902361.00	1726187.00
[3,]	0.83	0.7	0.61	0.54	0.48	0.28

```
> MEDIPS.plotCoverage(cr.treat)
```



```
> er.treat = MEDIPS.CpGenrich(data = TREAT.SET)
```

```
> er.treat
```

```
$regions.CG
```

```
[1] 5793412
```

```
$regions.C
```

```
[1] 60173176
```

```
$regions.G
```

```
[1] 60327187
```

```
$regions.relH
```

```
[1] 2.333744
```

```
$regions.GoGe
```

```
[1] 0.3961865
```

```
$genome.C
```

```
[1] 592445731
```

```
$genome.G
```

```
[1] 592804204
```

```
$genome.CG
```

```
[1] 28670425
```

```
$genome.relH
```

```
[1] 0.9903856
```

```
$genome.GoGe
```

```
[1] 0.2363220
```

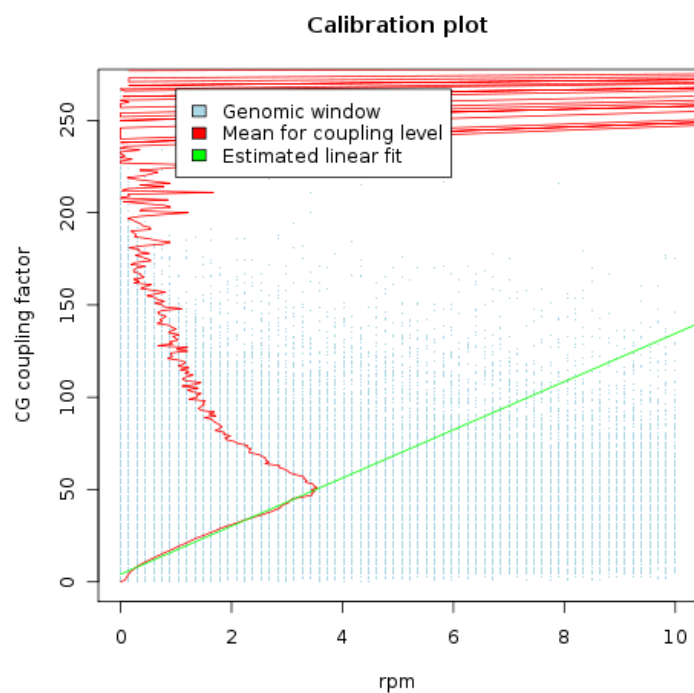
```
$enrichment.score.relH
```

```
[1] 2.356399
```

```
$enrichment.score.GoGe
```

```
[1] 1.676469
```

```
> MEDIPS.plotCalibrationPlot(data = TREAT.SET, linearFit = T, plot_chr = "chr1",  
+   rpm = T, xrange = 10)
```



```
> MEDIPS.exportWIG(file = "output.rpm.treat.WIG", data = TREAT.SET,  
+   raw = T, descr = "DE.rpm")
```



```
> MEDIPS.exportWIG(file = "output.rms.treat.WIG", data = TREAT.SET,
+   raw = F, descr = "DE.rms")
```

It is recommended (but not necessary) to provide background data from an INPUT experiment (that is sequencing of non-enriched DNA fragments). By providing an INPUT data set, MEDIPS will calculate a background data distribution used for the identification of DMRs. Therefore, we now process the available INPUT data within the file `Input_StemCells_chr1-3.txt`:

```
> INPUT.SET = MEDIPS.readAlignedSequences(BSgenome = "BSgenome.Hsapiens.UCSC.hg19",
+   file = "Input_StemCells_chr1-3.txt", numRows = 5226476)

> INPUT.SET = MEDIPS.genomeVector(data = INPUT.SET, bin_size = 50,
+   extend = 400)
```

For INPUT data, we do not correct for CpG densities because no enrichment for methylated CpG's was performed. Only the rpm values will be considered. Therefore, the INPUT MEDIPS SET is already sufficiently processed for being integrated into the process of identifying DMRs. Nevertheless, here we execute the quality controls for the INPUT set:

```
> sr.input = MEDIPS.saturationAnalysis(data = INPUT.SET, bin_size = 50,
+   extend = 400, no_iterations = 10, no_random_iterations = 1)
```

```
> sr.input
```

```
$distinctSets
```

	[,1]	[,2]
[1,]	0	0.0000000
[2,]	261323	0.1052366
[3,]	522646	0.1900837
[4,]	783969	0.2600781
[5,]	1045292	0.3237190
[6,]	1306615	0.3740161
[7,]	1567938	0.4158374
[8,]	1829261	0.4563268
[9,]	2090584	0.4897627
[10,]	2351907	0.5198877
[11,]	2613238	0.5481245

```
$estimation
```

	[,1]	[,2]
[1,]	0	0.0000000
[2,]	261323	0.1305343
[3,]	522646	0.2294398
[4,]	783969	0.3111776
[5,]	1045292	0.3744069
[6,]	1306615	0.4276966
[7,]	1567938	0.4716949
[8,]	1829261	0.5088855
[9,]	2090584	0.5416000

```
[10,] 2351907 0.5698553
[11,] 2613230 0.5954322
[12,] 2874553 0.6183213
[13,] 3135876 0.6378591
[14,] 3397199 0.6559742
[15,] 3658522 0.6731114
[16,] 3919845 0.6883246
[17,] 4181168 0.7019292
[18,] 4442491 0.7140006
[19,] 4703814 0.7248599
[20,] 4965137 0.7348504
[21,] 5226476 0.7447858
```

```
$numberReads
```

```
[1] 5226476
```

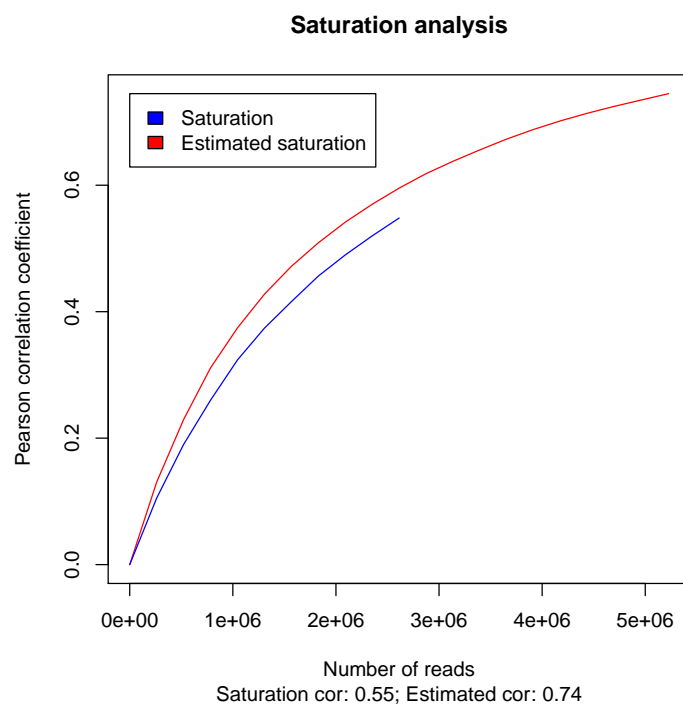
```
$maxEstCor
```

```
[1] 5.226476e+06 7.447858e-01
```

```
$maxTruCor
```

```
[1] 2.613238e+06 5.481245e-01
```

```
> MEDIPS.plotSaturation(sr.input)
```



```
> INPUT.SET = MEDIPS.getPositions(data = INPUT.SET, pattern = "CG")
> cr.input = MEDIPS.coverageAnalysis(data = INPUT.SET, extend = 400,
+   no_iterations = 10)
```

```

> cr.input

$matrix
      [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]
[1,]      0      1      2      3      4      5      10
[2,]      0      0      0      0      0      0      0
[3,] 522647 1774353 346723 54074 8241 1998 619
[4,] 1045294 2943260 1026759 289725 71188 16736 1227
[5,] 1567941 3743229 1768148 688578 234679 71972 1850
[6,] 2090588 4299118 2447627 1176219 498313 189472 2961
[7,] 2613235 4692394 3035008 1692013 839811 377450 5482
[8,] 3135882 4976625 3526465 2193059 1226386 626593 11386
[9,] 3658529 5186062 3930563 2653502 1629938 922833 23458
[10,] 4181176 5342541 4261498 3068684 2029307 1244726 46597
[11,] 4703823 5461164 4532058 3435355 2410997 1581365 84146
[12,] 5226476 5550467 4752851 3756417 2765023 1916799 140957

$maxPos
[1] 6072451

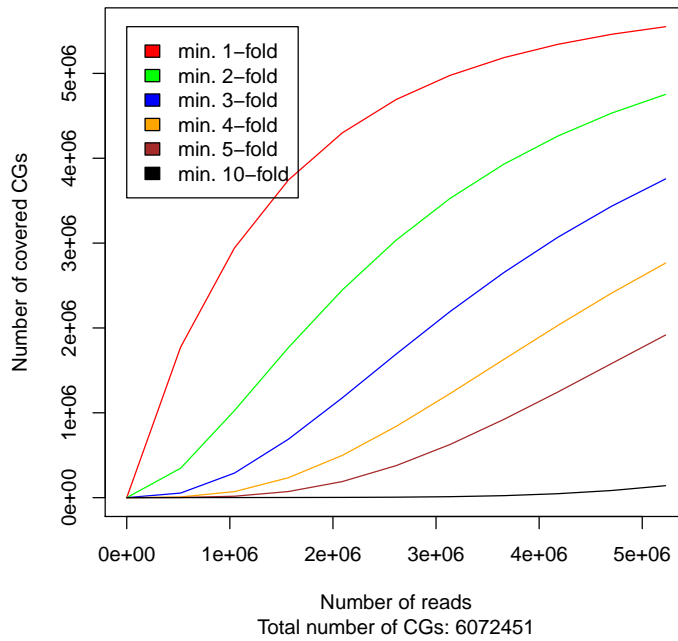
$pattern
[1] "CG"

$coveredPos
      [,1]      [,2]      [,3]      [,4]      [,5]      [,6]
[1,]      1.00      2.00      3.00      4.00      5.00     10.00
[2,] 5550467.00 4752851.00 3756417.00 2765023.00 1916799.00 140957.00
[3,]      0.91      0.78      0.62      0.46      0.32      0.02

> MEDIPS.plotCoverage(cr.input)

```

Coverage analysis



```
> er.input = MEDIPS.CpGenrich(data = INPUT.SET)
```

```
> er.input
```

```
$regions.CG
[1] 2042872
```

```
$regions.C
[1] 42772505
```

```
$regions.G
[1] 42933872
```

```
$regions.relH
[1] 1.056405
```

```
$regions.GoGe
[1] 0.2151231
```

```
$genome.C
[1] 592445731
```

```
$genome.G
[1] 592804204
```

```
$genome.CG
[1] 28670425
```

```

$genome.relH
[1] 0.9903856

$genome.GoGe
[1] 0.2363220

$enrichment.score.relH
[1] 1.066660

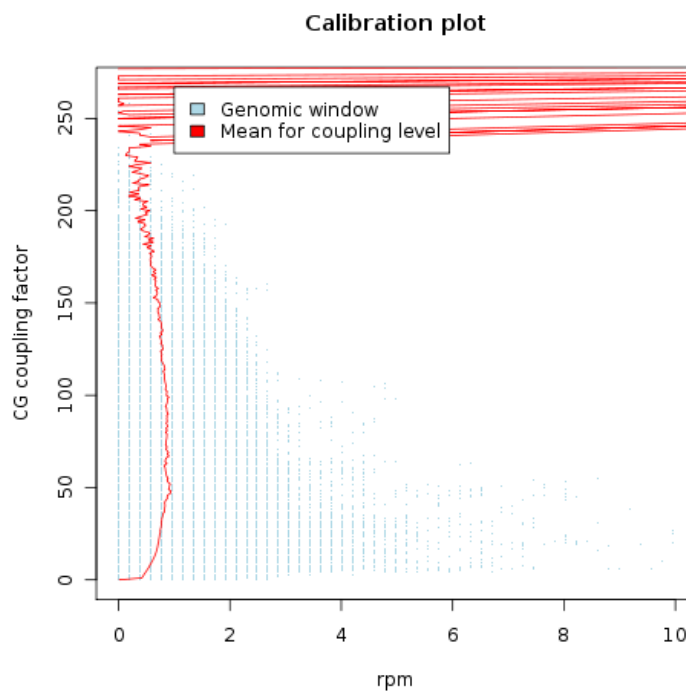
$enrichment.score.GoGe
[1] 0.9102968

> INPUT.SET = MEDIPS.couplingVector(data = INPUT.SET, fragmentLength = 700,
+   func = "count")

> INPUT.SET = MEDIPS.calibrationCurve(data = INPUT.SET)

> MEDIPS.plotCalibrationPlot(data = INPUT.SET, linearFit = F, plot_chr = "chr1",
+   rpm = T, xrange = 10)

```



```

> MEDIPS.exportWIG(file = "output.rpm.input.WIG", data = INPUT.SET,
+   raw = T, descr = "INPUT.rpm")

```

From the saturation analysis of the INPUT data, it can be deduced that in general, the correlation between the independent genome vectors increases

more slowly than for MeDIP data. This is due to an increased complexity of available genomic DNA that has to be sequenced when no specific enrichment was performed. The CpG enrichment values indicate that INPUT data is not enriched for CpGs. Additionally, the calibration curve of the calibration plot allows for discriminating MeDIP from INPUT data.

3.12 Methylation Profiles of Two MEDIPS SET Objects

As we have generated two MEDIPS SETs and one INPUT SET, we will now calculate mean methylation values for all three sets in parallel, and compare the methylation signals between the sets. Methylation profiles and comparisons can be performed for given regions of interest (see section 3.10) or for genome wide (sliding) frames. This is all done by the `MEDIPS.methylProfiling()` function.

Here, we process genome wide overlapping 500bp windows:

```
> diff.meth = MEDIPS.methylProfiling(data1 = CONTROL.SET, data2 = TREAT.SET,
+   input = INPUT.SET, frame_size = 500, step = 250, select = 2)
```

Please note, the `MEDIPS.methylProfiling()` function takes a comparatively long processing time when called for genome wide short windows. In fact, this is the most time-consuming bottleneck of the whole procedure for identifying DMRs.

The results are stored as a list at the specified object (here `diff.meth`). All list objects are vectors of the same length, where the length is defined by the number of tested frames. Each row refers to a ROI and the individual vectors are as described in section 3.10. Here, all vectors are occupied, including mean methylation values for the two MEDIPS SETs and for the INPUT SET, as well as ratios and p-values obtained from comparing the two MEDIPS SETs.

While the `MEDIPS.methylProfiling()` function is running, it is again time for some theory. Let C , T , and I be the genome vectors generated based on the sequencing data from the `CONTROL.SET`, `TREAT.SET`, and `INPUT.SET` objects using an arbitrary bin size b and let ROI be a set of ROIs (e.g. genome wide windows), where $ROI = ROI_1, \dots, ROI_i, \dots, ROI_n$, and n is the number of ROIs to be tested. Let the ROIs be of length m_1, \dots, m_n . In the following, the identification of DMRs is only supported for any ROI_i of length $m_i \geq 5 \cdot b$. Therefore, each ROI_i includes at least five genomic bins $bin_{i,j}$, where $bin_{i,1}, \dots, bin_{i,j}, \dots, bin_{i,k_i} \in ROI_i$ and $k_i = \text{floor}(\frac{m_i}{b})$. For each ROI_i , mean rpm and mean rms values are calculated based on C and T as:

$$C.RPM_{ROI_i} = \frac{1}{k_i} \sum_{j=1}^{k_i} rpm(C.b_{i,j})$$

$$C.RMS_{ROI_i} = \frac{1}{k_i} \sum_{j=1}^{k_i} rms(C.b_{i,j})$$

$$T.RPM_{ROI_i} = \frac{1}{k_i} \sum_{j=1}^{k_i} rpm(T.b_{i,j})$$

$$T.RMS_{ROI_i} = \frac{1}{k_i} \sum_{j=1}^{k_i} rms(T.b_{i,j})$$

where $rpm(C.b_{i,j})$, $rms(C.b_{i,j})$, $rpm(T.b_{i,j})$, and $rms(T.b_{i,j})$ are the pre-calculated rpm and rms (see sections 3.2 and 3.9) values of the genomic bins from the Control and Treatment samples. In addition, for each ROI_i , mean rpm values are calculated based on I as:

$$I.RPM_{ROI_i} = \frac{1}{k_i} \sum_{j=1}^{k_i} rpm(I.b_{i,j})$$

where $rpm(I.b_{i,j})$ are the pre-calculated rpm values of the genomic bins from the Input sample. Based on the mean rms values of the Control and of the Treatment sample, for each ROI_i the following ratios are calculated:

$$r.rms_{ROI_i} = \frac{C.RMS_{ROI_i}}{T.RMS_{ROI_i}}$$

In addition, by considering the mean rpm values of the Control or of the Treatment sample, respectively, the following ratios are calculated with respect to mean rpm values of the Input sample:

$$r.rpm.C_{ROI_i} = \frac{C.RPM_{ROI_i}}{I.RPM_{ROI_i}}$$

$$r.rpm.T_{ROI_i} = \frac{T.RPM_{ROI_i}}{I.RPM_{ROI_i}}$$

Because local background sequencing signals are variable along the chromosomes due to differing DNA availability, a global background rpm signal threshold is estimated based on the distribution of all calculated $I.RPM_{ROI_i}$ values. This is done by defining a targeted quantile qt (e.g. $qt = 0.95$) and by identifying the $I.RPM_{ROI_i}$ value (t), where $qt\%$ of all $I.RPM_{ROI_i}$ values are $< t$. This estimated global minimal mean rpm threshold t will serve as an additional parameter for selecting genomic regions that show a mean MedIP-Seq derived rpm signal of at least t in the Control or the Treatment sample, respectively (see next section).

In addition, statistical testing is utilized in order to rate whether the obtained rms data series of the genomic bins within any ROI_i significantly differ in the Control sample compared to the Treatment sample. For each ROI_i it is tested, whether the rms values of the genomic bins $bin_{i,1}, \dots, bin_{i,j}, \dots, bin_{i,k_i} \in ROI_i$ of the Control sample significantly differ from the rms values of the according genomic bins of the Treatment sample. For this, the MEDIPS package utilizes the `t.test()` and `wilcox.test()` functions of the R environment with default parameter settings (two-sided tests in both cases). Therefore, for each tested ROI_i , two p-values ($ROI.p.value.t_i$ and $ROI.p.value.w_i$) will be calculated and serve as a further level for discriminating between local methylation profiles (see next section).

Please note, the returned mean rms and ams within the *diff.meth* object are in log2 scale and were transformed into the consistent interval [0:1000], but ratios and p-values are calculated based on the non-transformed mean values. Moreover, ratios and p-values can be calculated based on the rpm values (instead of rms values) by specifying the parameter `select=1`, when calling the `MEDIPS.methylProfiling()` function.

You may want to store the received results matrix by using R's `write.table()` function like:

```
> write.table(diff.meth, file = "diff.meth.txt", sep = "\t", quote = F,
+           col.names = T, row.names = F)
```

You can upload the results matrix at any later time into R by typing

```
> diff.meth = read.table(file = "diff.meth.txt", header = T)
```

3.13 Selecting Candidate DMRs and Annotation

Based on the results returned from the `MEDIPS.methylProfiling()` function in section 3.12 (here `diff.meth`), we now select candidate ROIs that show significant differential methylation between the CONTROL.SET and the TREAT.SET in consideration of the background data included in the INPUT.SET. For this, MEDIPS offers the possibility to specify the following parameters in order to apply several filters to the full set of ROIs:

- **frames:** specifies the results table
- **input:** default=T; Setting the parameter to TRUE requires that the results table includes a column for summarized *rpm* values of an INPUT SET. In case there is no INPUT data available, the **input** parameter has to be set to a *rpm* value that will be used as a lower threshold during the subsequent analysis. How to estimate such a threshold without background data is not yet solved by MEDIPS.
- **quant:** default=0.9; from the distribution of all summarized INPUT *rpm* values, MEDIPS calculates the *rpm* value that represents the **quant** quantile of the whole INPUT distribution.
- **control:** can be either TRUE or FALSE. MEDIPS allows for selecting ROIs that are higher methylated in the CONTROL SET compared to the TREAT SET and vice versa but both approaches have to be performed in two independent runs. By setting **control=TRUE**, ROIs will be selected that are higher methylated in the CONTROL SET. By setting **control=FALSE**, ROIs will be selected that are higher methylated in the TREAT SET.
- **up:** default=1.333333; defines the lower threshold for the ratio CONTROL/TREATMENT as well as the lower ratio for CONTROL/INPUT (if **control=T**) or TREATMENT/INPUT (if **control=F**), respectively.
- **down:** default=0.75; defines the upper threshold for the ratio: CONTROL/TREATMENT (only if **control=F**).
- **p.value:** default=0.01; defines the threshold for the p-values. One of the two p-values derived from the `wilcox.test` and `t.test` function has to be \leq **p.value**.

The following command filters for candidate frames that are higher methylated in human embryonic stem cells than in differentiated hESCs:

```
> diff.meth.control = MEDIPS.selectSignificants(frames = diff.meth,
+           control = T, input = T, up = 2, p.value = 1e-04, quant = 0.99)
```



```

[1] Processing input distribution...
[1] Done.
[1] Total number of frames: 2761891
[1] Number of frames where control or treatment !=0: 2463305
[1] Remaining number of frames with p.value<=1e-04: 275967
[1] Remaining number of frames where control/treatment ratio >= 2: 147137
[1] Estimated rpm threshold for input quantile 0.99 is: 1.68373489134935
[1] Remaining number of frames with control rpm >=1.68373489134935: 17894
[1] Remaining number of frames with control/input ratio>=2: 14559
[1] [Note: There are 0 frames associated to a p-value==0.]
[1] [Note: There are 6 frames, where control/treatment ratio = Inf (i.e. treatment==0).]

```

For identifying ROI_i 's that show differential methylation between the Control and the Treatment sample with respect to the Input sample, based on the pre-calculated parameters (see previous section), a filtering procedure is performed. The following filtering procedure also discriminates between increased methylation in the Control sample compared to the Treatment sample (Control>Treatment, a) and vice versa (Treatment>Control, b):

1. ROI_i 's where $C.RPM_{ROI_i} = T.RPM_{ROI_i} = 0$ are neglected,
2. ROI_i 's where $ROI.p.value.t_i > p.value$ and $ROI.p.value.w_i > p$ are neglected, where $p.value$ is any targeted level of significance (e.g. $p.value = 0.01$),
3. Filtering for the ratio:
 - a) ROI_i 's where $r.rms_{ROI_i} < up$ are neglected, where up is an upper ratio threshold (e.g. $up = 1.33$),
 - b) ROI_i 's where $r.rms_{ROI_i} > down$ are neglected, where $down$ is a lower ratio threshold (e.g. $down = 0.75$),
4. Filtering for global Input derived background signals:
 - a) ROI_i 's where $C.RPM_{ROI_i} < t$ are neglected,
 - b) ROI_i 's where $T.RPM_{ROI_i} < t$ are neglected,
5. Filtering for local Input derived background signals:
 - a) ROI_i 's where $r.rpm.C_{ROI_i} < up$ are neglected,
 - b) ROI_i 's where $r.rpm.T_{ROI_i} < up$ are neglected.

In our example, there are 14,559 frames of length 500bp that remain after the `MEDIPS.selectSignificants()` step. Because we have executed the `MEDIPS.methylProfiling()` function by setting `frame_size=500` and `step=250`, we may have overlapping significant frames stored in our `diff.meth.control` results table. Whenever for the parameter settings it is valid `step<frame_size`, we may end up with overlapping significant frames. For these cases it is worthwhile to merge overlapping regions by typing:

```
> diff.meth.control.merged = MEDIPS.mergeFrames(frames = diff.meth.control)
```

```
[1] Number of merged frames: 10375
```

Please note, the remaining 10,375 distinct frames are represented only by their genomic coordinates within the `diff.meth.control.merged` table (these are the chromosome names, new start, and new stop positions). The results table does not contain any merged *rpm*, *rms*, *variance*, *p.value*, etc. values any more.

Moreover, it is important to keep in mind, that there are three main reasons why an analysis of only a subset of the full genome (here only chromosomes 1, 2, and 3) will probably end up in a different number of significant DMRs: First, the calibration parameters were calculated for only three chromosomes. Second, the total number of given regions differs and therefore, the reads per million values will differ. Third, the *rpm* threshold derived from the background INPUT data distribution was calculated based on data from only three chromosomes.

However, you can write out the obtained regions by typing some suitable R code like:

```
> write.table(diff.meth.control.merged, file = "DiffMethyl.Up.hESCs.bed",
+           sep = "\t", quote = F, row.names = F, col.names = F)
```

You can upload the resulting file into a genome browser and the DMRs will be visualized as black blocks.

Finally, it might be of interest to annoate the DMRs. We fall back on the example ROI file `hg19.chr1-3.txt` that contains pre-defined promoter regions (-1kb to +0.5kb around the TSSs). We annotate the identified DMRs by the transcript promoters included in the ROI file by typing

```
> diff.meth.control.merged.annotated = MEDIPS.annotate(diff.meth.control.merged,
+           anno = "hg19.chr1-3.txt")
```

The resulting table is of the following format:

1. **chr:** the chromosome name of the DMR
2. **start:** the start position of the DMR
3. **end:** the stop position of the DMR
4. **feature:** the name of the annotation

For each provided region (DMR), the function returns all overlapping annotations included in the provided annotation file. Note, in case there are several overlapping annotations, the region (DMR) is returned several times in sparated rows, each entry associated to one annotation. In order to receive e.g. a unique list of ensembl transcript names whose promoter regions overlap with a DMR, you can now easily select for the appropriate entries using standard R commands, e.g.

```
> length(unique(diff.meth.control.merged.annotated[, 4]))
```

```
[1] 714
```

In our example, the 10,375 non-overlapping identified DMRs can be associated to promoter regions of 714 unique transcripts (including genes as well as pseudogenes and small RNAs). These DMRs can be interpreted as regions

where de-methylation events occur during the differentiation of hESCs along the endodermal lineage.

The other way round, we now select for frames higher methylated in differentiated hESCs (DE) than in pluripotent hESCs:

```
> diff.meth.treat = MEDIPS.selectSignificants(frames = diff.meth,
+       control = F, input = T, up = 2, down = 0.5, p.value = 1e-04,
+       quant = 0.99)

[1] Processing input distribution...
[1] Done.
[1] Total number of frames: 2761891
[1] Number of frames where control or treatment !=0: 2463305
[1] Remaining number of frames with p.value<=1e-04: 275967
[1] Remaining number of frames where treatment/control ratio <= 0.5: 87040
[1] Estimated rpm threshold for input quantile 0.99 is: 1.68373489134935
[1] Remaining number of frames with treatment rpm >=1.68373489134935: 4413
[1] Remaining number of frames with treatment vs. input ratio>=2: 3559
[1] [Note: There are 0 frames associated to a p-value==0.]
[1] [Note: There are 10 frames, where control/treatment ratio = 0 (i.e. control=0).]

> diff.meth.treat.merged = MEDIPS.mergeFrames(diff.meth.treat)

[1] Number of merged frames: 3035

> write.table(diff.meth.treat.merged, file = "DiffMethyl.Up.DE.bed",
+       sep = "\t", quote = F, row.names = F, col.names = F)

> diff.meth.treat.merged.annotated = MEDIPS.annotate(diff.meth.treat.merged,
+       anno = "hg19.chr1-3.txt")

> length(unique(diff.meth.treat.merged.annotated[, 4]))

[1] 893
```

In our example, there are 3,035 non-overlapping DMRs associated the promoter regions of 893 unique transcripts. These DMRs can be interpreted as regions where a **denovo** methylation events occur during the differentiation of hESCs along the endodermal lineage.

4 Concluding Remarks

In our opinion, MEDIPS provides several helpful functionalities for analysing MeDIP-Seq data in reasonable time compared to other available approaches. Nevertheless, there are some limitations that have to be addressed in the future. The main issues are:

- Because MEDIPS processes full genome data at once, MEDIPS needs a lot of memory. Especially, when two MEDIPS SETs as well as an INPUT SET is uploaded and utilized for the identification of DMRs, the need for memory is very huge.

- The runtime, especially the calculation of mean methylation values or DMRs for genome wide short windows remains a bottleneck.
- The `MEDIPS.methylProfiling()` function is a novel approach for identifying enriched regions, and is especially suitable for MeDIP-Seq data because DNA methylation is expected to occur on longer DNA stretches compared to smaller enrichments derived from e.g. ChIP-Seq data. However, the identification of DMRs is very static. The definition of fixed windows, although when overlapping windows are defined, is not very flexible.

However, to the best of our knowledge, MEDIPS is currently the most comprehensive software for processing MeDIP-Seq data. It starts where the mapping tools stop, touches several aspects of data quality checks, allows for exporting raw and normalized methylation profiles, calculates mean methylation values for any specified ROI and identifies genome wide DMRs when hunting for differential DNA-methylation comparing two conditions.

References

- Lukas Chavez, Justyna Jozefczuk, Christina Grimm, Joern Dietrich, Bernd Timmermann, Ralf Herwig, and James Adjaye. Computational analysis of genome-wide dna-methylation during the differentiation of human embryonic stem cells along the endodermal lineage. *in preparation*, 2010.
- Thomas A Down, Vardhman K Rakyan, Daniel J Turner, Paul Flicek, Heng Li, Eugene Kulesha, Stefan Graef, Nathan Johnson, Javier Herrero, Eleni M Tomazou, Natalie P Thorne, Liselotte Baeckdahl, Marlis Herberth, Kevin L Howe, David K Jackson, Marcos M Miretti, John C Marioni, Ewan Birney, Tim J P Hubbard, Richard Durbin, Simon Tavare, and Stephan Beck. A bayesian deconvolution strategy for immunoprecipitation-based dna methylome analysis. *Nat Biotechnol*, 26(7):779–785, Jul 2008.
- Florian Eckhardt, Joern Lewin, Rene Cortese, Vardhman K Rakyan, John Attwood, Matthias Burger, John Burton, Tony V Cox, Rob Davies, Thomas A Down, Carolina Haefliger, Roger Horton, Kevin Howe, David K Jackson, Jan Kunde, Christoph Koenig, Jennifer Liddle, David Niblett, Thomas Otto, Roger Pettett, Stefanie Seemann, Christian Thompson, Tony West, Jane Rogers, Alex Olek, Kurt Berlin, and Stephan Beck. Dna methylation profiling of human chromosomes 6, 20 and 22. *Nat Genet*, 38(12):1378–1385, Dec 2006.
- Ryan Lister, Mattia Pelizzola, Robert H Dowen, R. David Hawkins, Gary Hon, Julian Tonti-Filippini, Joseph R Nery, Leonard Lee, Zhen Ye, Que-Minh Ngo, Lee Edsall, Jessica Antosiewicz-Bourget, Ron Stewart, Victor Ruotti, A. Harvey Millar, James A Thomson, Bing Ren, and Joseph R Ecker. Human dna methylomes at base resolution show widespread epigenomic differences. *Nature*, 462(7271):315–322, Nov 2009.
- Mattia Pelizzola, Yasuo Koga, Alexander Eckehart Urban, Michael Krauthammer, Sherman Weissman, Ruth Halaban, and Annette M Molinaro. Medme:

an experimental and analytical methodology for the estimation of dna methylation levels based on microarray derived medip-enrichment. *Genome Res*, 18(10):1652–1659, Oct 2008.

Michael Weber, Jonathan J Davies, David Wittig, Edward J Oakeley, Michael Haase, Wan L Lam, and Dirk Schuebeler. Chromosome-wide and promoter-specific analyses identify sites of differential dna methylation in normal and transformed human cells. *Nat Genet*, 37(8):853–862, Aug 2005.