MEDIPS Tutorial

Lukas Chavez

March 30, 2012

Contents

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, MEDIPS may be applied to other immunoprecipitation based methylation analyses (e.g. MBD-seq), and selected functionalities like the saturation analysis may be applied to other types of sequencing data (e.g. ChIP-seq). MEDIPS adresses 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 immuno-precipitation,

- 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 and with respect to Input data (if available), and

- 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 the Bioconductor `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, there is a bug in R distributions $<=2.10.1$ that sometimes causes errors when interacting with the BSgenome package. This bug should be fixed in R versions $>2.11.0$.

Next, it is necessary to install the MEDIPS package into your R environment. Start R and enter:

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

Please note, MEDIPS became available on BioC 2.7 that is designed to work with R.2.12. Therefore, installation of the MEDIPS package by `biocLite()` only works on R >=2.12.0.

In order to reproduce the presented MEDIPS workflow, the package includes the example data sets `MeDIP_hESCs_chr22.txt` (17M), `MeDIP_DE_chr22.txt` (22M), and `Input_StemCells_chr22.txt` (8.8M) in the `extdata` subdirectory of the MEDIPS package.

The files contain genomic regions from chromosome 22 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 from hESCs and DE [Chavez et al., 2010].

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

- the first coulumn 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 artifical 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.Alyrata.JGI.v1"
 [2] "BSgenome.Amellifera.BeeBase.assembly4"
 [3] "BSgenome.Amellifera.UCSC.apiMel2"
 [4] "BSgenome.Athaliana.TAIR.04232008"
 [5] "BSgenome.Athaliana.TAIR.TAIR9"
```

```
 [6] "BSgenome.Btaurus.UCSC.bosTau3"
 [7] "BSgenome.Btaurus.UCSC.bosTau4"
 [8] "BSgenome.Celegans.UCSC.ce2"
 [9] "BSgenome.Celegans.UCSC.ce6"
[10] "BSgenome.Cfamiliaris.UCSC.canFam2"
[11] "BSgenome.Dmelanogaster.UCSC.dm2"
[12] "BSgenome.Dmelanogaster.UCSC.dm3"
[13] "BSgenome.Drerio.UCSC.danRer5"
[14] "BSgenome.Drerio.UCSC.danRer6"
[15] "BSgenome.Drerio.UCSC.danRer7"
[16] "BSgenome.Ecoli.NCBI.20080805"
[17] "BSgenome.Gaculeatus.UCSC.gasAcu1"
[18] "BSgenome.Ggallus.UCSC.galGal3"
[19] "BSgenome.Hsapiens.UCSC.hg17"
[20] "BSgenome.Hsapiens.UCSC.hg18"
[21] "BSgenome.Hsapiens.UCSC.hg19"
[22] "BSgenome.Mmulatta.UCSC.rheMac2"
[23] "BSgenome.Mmusculus.UCSC.mm10"
[24] "BSgenome.Mmusculus.UCSC.mm8"
[25] "BSgenome.Mmusculus.UCSC.mm9"
[26] "BSgenome.Ptroglodytes.UCSC.panTro2"
[27] "BSgenome.Rnorvegicus.UCSC.rn4"
[28] "BSgenome.Scerevisiae.UCSC.sacCer1"
[29] "BSgenome.Scerevisiae.UCSC.sacCer2"
[30] "BSgenome.Scerevisiae.UCSC.sacCer3"
[31] "BSgenome.Tgondii.ToxoDB.7.0"
```

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.

```
> 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. This is performed by the `MEDIPS.readAlignedSequences()` function. When calling this function, it is necessary to state the reference genome build and your input file. In case you know the number of rows within your input file, you can also specify the `numrows` parameter. This will accelerate the reading of your file. Otherwise just skip the `numrows` parameter.

In this manual, we are using example data located in the `extdata` subdirectory of the MEDIPS package. Therefore, we have to point to the input example files like:

```
> file=system.file("extdata", "MeDIP_hESCs_chr22.txt", package="MEDIPS")
```

In the example `MeDIP_hESCs_chr22.txt` file, there are 672866 regions (here, only chromosome 22 is included). Therefore, we can now read the example file by typing:

```
> CONTROL.SET=MEDIPS.readAlignedSequences(BSgenome="BSgenome.Hsapiens.UCSC.hg19", file=fil
```

Typically, you will execute the function by pointing directly to your input file using the `file` parameter. Here, you can also state the full directory path to your input file together with the file name.

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 information
======================================
Regions file:  MeDIP_hESCs_chr22.txt
Organism:  BSgenome.Hsapiens.UCSC.hg19
Chromosomes: chr22
Chromosome lengths: 51304566
Number of regions:  672866
Regions chromosomes: chr22 chr22 chr22...
Regions start positions: 16053184 16054710 16080510...
Regions stop positions: 16053220 16054746 16080546...
Regions strand: - - +...
======================================
Genome vector signals information
======================================
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 information
======================================
```

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

After reading the input file, the MEDIPS SET contains only 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,...,m` and `m` is the total number of genomic bins, then the `rpm` value of the genomic bin is simply defined as:

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

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.rp
```

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 insuffcient 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_itera
```

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 336433 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` ($A_1, A_2, .., A_{10}$) and of `B` ($B_1, B_2, .., B_{10}$) will contain approx. 33643 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 examin 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 this artificially doubled set of regions (see also Supplementary Methods in [Chavez et al., 2010]).

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,]   33643 0.4407305
 [3,]   67286 0.6136978
 [4,]  100929 0.7058264
 [5,]  134572 0.7633364
 [6,]  168215 0.8017750
 [7,]  201858 0.8297277
 [8,]  235501 0.8495057
 [9,]  269144 0.8650050
[10,]  302787 0.8785235
[11,]  336433 0.8891640

$estimation
         [,1]      [,2]
 [1,]       0 0.0000000
```

```
 [2,]  33643 0.4511955
 [3,]  67286 0.6221709
 [4,] 100929 0.7128588
 [5,] 134572 0.7680349
 [6,] 168215 0.8057714
 [7,] 201858 0.8328346
 [8,] 235501 0.8532486
 [9,] 269144 0.8692397
[10,] 302787 0.8819233
[11,] 336430 0.8925810
[12,] 370073 0.9013703
[13,] 403716 0.9086962
[14,] 437359 0.9149013
[15,] 471002 0.9201913
[16,] 504645 0.9250774
[17,] 538288 0.9294995
[18,] 571931 0.9332933
[19,] 605574 0.9367776
[20,] 639217 0.9400645
[21,] 672866 0.9427153


$numberReads
[1] 672866


$maxEstCor
[1] 6.728660e+05 9.427153e-01


$maxTruCor
[1] 3.36433e+05 8.89164e-01
```
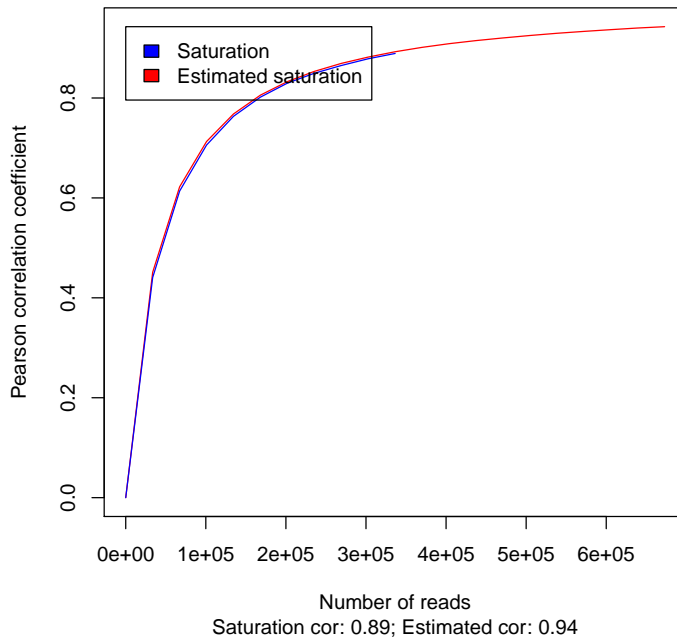
The maximal obtained correlation of the saturation analysis is stored at the maxTruCor slot and the maximal obtained correlation of the estimated saturation analysis is stored at the maxEstCor slot of the saturation results object (first column: total number of considered reads, second column: obtained 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

Saturation cor: 0.89; Estimated cor: 0.94

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 (0.89) and of the estimated (0.94) saturation analyis.

The saturation analysis assists for rating whether the costs of additional sequencing runs are in proportion to the gain in 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 chromosome 22 is considered).

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

- `no_random_iterations`: methods 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 all the random iteration steps.

- `empty_bins`: can be either TRUE or FALSE (default TRUE). This parameter affects the way of calculating correlations between the resulting genome vectors. 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 paramter 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 occuring 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 cytosins. For this, an antibody is used that recognizes 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). Afterwards, 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 distinct 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 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,]  67286 290011 154223  82599  44626  24683   1647
 [4,] 134572 379227 262668 182054 125805  88153  16027
 [5,] 201858 422229 326535 251468 193507 149214  43876
 [6,] 269144 447673 367612 301036 245194 199656  75423
 [7,] 336430 464545 395438 336370 285611 241454 106340
 [8,] 403716 476488 415955 363528 316619 275562 136254
 [9,] 471002 485433 431447 384306 341121 302633 163885
[10,] 538288 492119 443591 400651 361313 325085 188324
[11,] 605574 497695 453251 414248 378216 344190 211410
[12,] 672866 502169 460963 425535 392066 360240 232422


$maxPos
[1] 578097


$pattern
[1] "CG"


$coveredPos
           [,1]      [,2]      [,3]      [,4]      [,5]      [,6]
[1,]       1.00       2.0      3.00      4.00      5.00      10.0
[2,] 502169.00 460963.0 425535.00 392066.00 360240.00 232422.0
[3,]       0.87       0.8      0.74      0.68      0.62       0.4
```
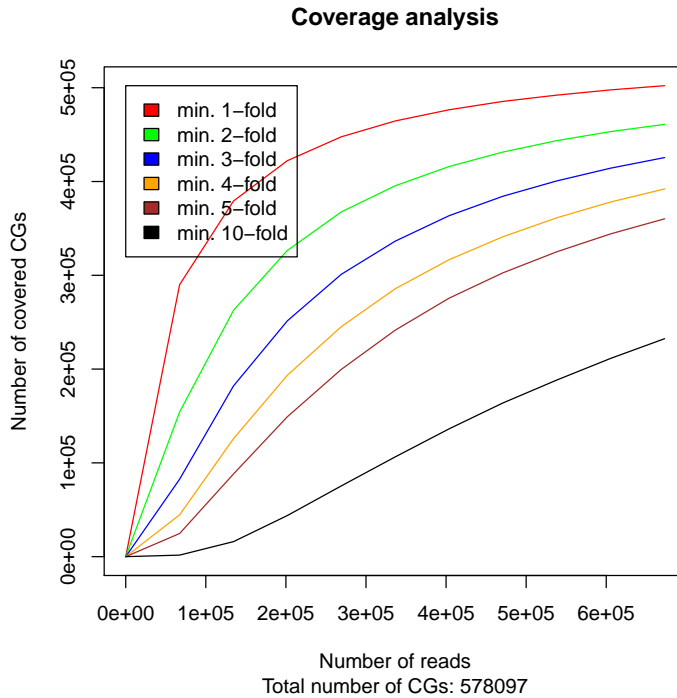
For example, the `maxPos` slot shows the total number of CpGs within chromosome 22 of the reference genome. Moreover, the `matrix` slot shows the results of the coverage analysis for each iteration and for each of the tested coverage depths. Here, the first row shows the tested levels of coverage. The 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)
```

Coverage analysis

Number of covered CGs

- min. 1–fold
- min. 2–fold
- min. 3–fold
- min. 4–fold
- min. 5–fold
- min. 10–fold

Number of reads
Total number of CGs: 578097

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

- `no_random_iterations`: methods 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 available short reads. These data quality controls assist in deciding whether the costs of additional experimental runs are in due proportion to the expected improvement 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 of Gs, 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] 628173

$regions.C
[1] 6551995

$regions.G
[1] 6575462

$regions.relH
[1] 2.523184

$regions.GoGe
[1] 0.3630026

$genome.C
[1] 592445731

$genome.G
[1] 592804204

$genome.CG
[1] 28670425

$genome.relH
[1] 0.9903856

$genome.GoGe
[1] 0.236322

$enrichment.score.relH
```

```
[1] 2.547679

$enrichment.score.GoGe
[1] 1.536051
```

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 Input experiments (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 2.547679, 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 dirction (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 by a varying efficiency of antibody binding with respect to local CpG densities. 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, first a coupling vector has to be calculated. The coupling vector is of the same size as the pre-defined 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 excecute 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 the 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 the following weighting functions (specified by the parameter `func`):

- `count`: simply count the number of CpGs within the predefined maximal distance to the current bin

- `linear`: the weights for CpGs decrease in a linear way and end at 0 at the predefined maximal distance to the current bin

- `exp`: the weights for CpGs decrease in an exponential way [Pelizzola et al., 2008]

- `log`: the weights for CpGs decrease 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. Based on their results, we have created the file `flat_400_700.tab` that can be downloaded from `http://medips.molgen.mpg.de`. You can create 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, desc
```

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.
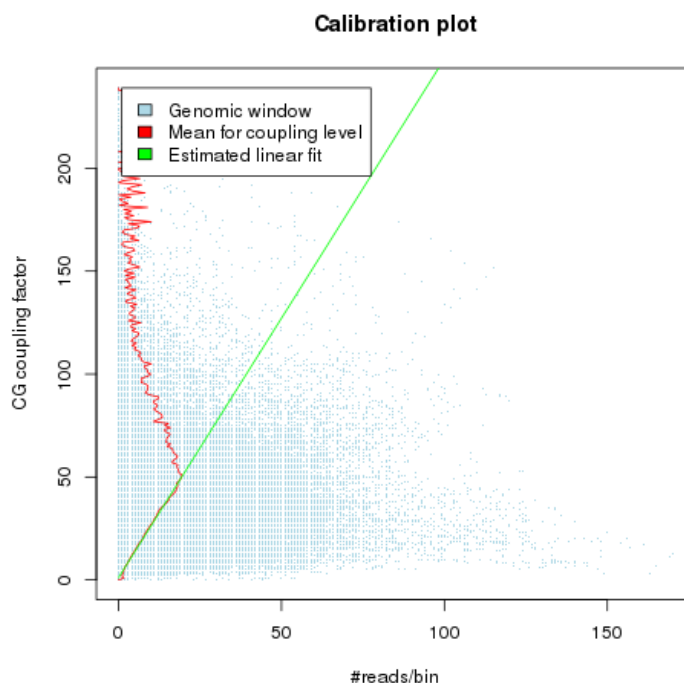
3.8 Calibration Curve and Linear Regression

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="chr22")
```

Calibration plot

Because the amount of data to be plotted can become very huge when plotting full genome data, it is strongly recommended to direct the output to a compressed file. This can be achieved by calling a e.g. `png("plot.png")` function before calling the 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

17

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. However, 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

$$y = y1, \ldots, yl$$

be the mean coupling factors, and let

$$x = x1, \ldots, xl$$

be the according mean MeDIP-seq signals of the calibration curve, where `l` is the number of tested coupling factor levels and `i = 1,...,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

$$y_{max} = y_1, ..., y_{i_{max}}$$

$$x_{max} = x_1, ..., x_{i_{max}}$$

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 $i = 1, ..., i_{max}$. 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,

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

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 $i = 1, ..., m$ 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.rm
```

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 information
======================================
Regions file:  MeDIP_hESCs_chr22.txt
Organism:  BSgenome.Hsapiens.UCSC.hg19
Chromosomes: chr22
Chromosome lengths: 51304566
Number of regions:  672866
Regions chromosomes: chr22 chr22 chr22...
Regions start positions: 16053184 16054710 16080510...
```

```
Regions stop positions: 16053220 16054746 16080546...
Regions strand: - - +...
=====================================
Genome vector signals information
=====================================
Genome wide bin size:  50
Reads extended by:  400
Genome vector chromosomes: chr22 chr22 chr22...
Genome vector positions: 1 51 101...
Genome vector signals: 0 0 0...
=====================================
Pattern information
=====================================
Pattern:  CG
Number of patterns:  578097
Pattern chromosomes: chr22 chr22 chr22...
Pattern positions: 16050097 16050114 16050174...
=====================================
Genome vector coupling factor information
=====================================
Distance function:  count
Distance file:  empty
Fragment length:  700
Genome vector coupling factors: 0 0 0...
=====================================
Calibration information
=====================================
Calibration curve mean signals: 0.001075285 1.609182 1.049421...
Calibration curve mean coupling factors: 0 1 2...
Calibration curve variance: NA NA NA...
Intercept:  1.15257423739513
Slope:  2.51962252361513
Calibration chromosome:  all
=====================================
Genome vector normalized signal information
=====================================
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 access the file `hg19.chr22.txt` in the `extdata` subdirectory of the MEDIPS package.

The file contains hg19 promoter regions of Ensembl transcripts located on the chromosome 22 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 summarized based on the generated MEDIPS SET object by typing:

```
> file=system.file("extdata", "hg19.chr22.txt", package="MEDIPS")
> promoter = MEDIPS.methylProfiling(data1 = CONTROL.SET, ROI_file = file, math = mean, sel
```

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.

- `transf`: default=`TRUE`; If set to TRUE, MEDIPS transforms the mean rms and ams values into log2 scale and subsequently transforms their resulting data range into the consistent interval $[0, 1000]$ before finally stored.

- `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 methylated cytosine specific 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. 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 set 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 generated and specified only one MEDIPS SET object so far, the results object only contains results (e.g. mean rpm and rms values) from 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 SET at `input` (if 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 at `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, see below) mean absolute methylation score (see below) for the MEDIPS SET at `data1`

12. `ams_B`: the (transformed, see below) 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, respectively, please see the parameter `select`) values of the MEDIPS SET at `data1`

14. `var_B`: the variance of the rpm (or rms, respectively, 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 variation of the rpm (or rms, respectively, please see the parameter `select`) of the MEDIPS SET at `data1`

16. `var_co_B`: the coefficient of variation of the rpm (or rms, respectively, 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)

Mean `rms` values (`rms_A` and `rms_B`) are calculated based on the pre-calculated rms_{bin_i} values of the genomic bins enclosed by the ROI. In case, the parameter $transf$ was set to TRUE, MEDIPS transforms the 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 always be 0 and the maximal transformed mean `rms_A` (or mean `rms_B`, respectively) will always 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 ROIs associated to e.g. `rms_A>600`. The other way round, lowly methylated ROIs may be subsequently identified by selecting ROIs associated to e.g. `rms_A<400`. However, it has to be kept in mind that the resulting data range ($[0, 1000]$) reflects the relative methylation levels of the tested ROIs.

We consider the rms values as experiment specific normalized MeDIP-seq signals corrected for the varying efficiency of antibody binding and immunoprecipitation in genomic regions with different CpG densities. 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 general CpG density of the region of interest and therefore, allow for comparing methylation profiles of genomic regions with 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 may influence 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 with different CpG densities.
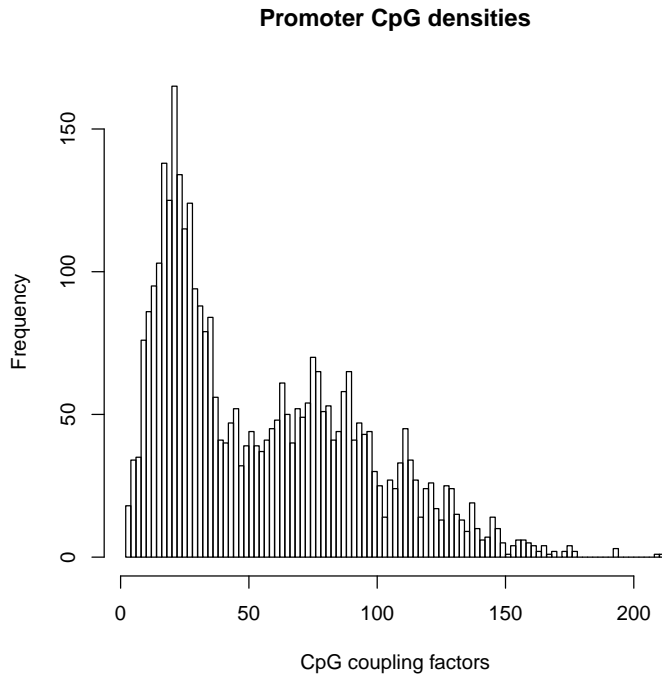
Let

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

be the raw MeDIP-seq signals and coupling factors of adjacent genomic bins i that define a region of interest (ROI), where $i = 1, ..., s$ 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 exectuted the $MEDIPS.methylProfiling()$ function for the specified ROI file containing promoter regions, one can have a look at e.g. the histogram of CpG densities or methylation levels. Simply call Rs `hist()` function and specify the desired column of the matrix. For the mean coupling factors type e.g.
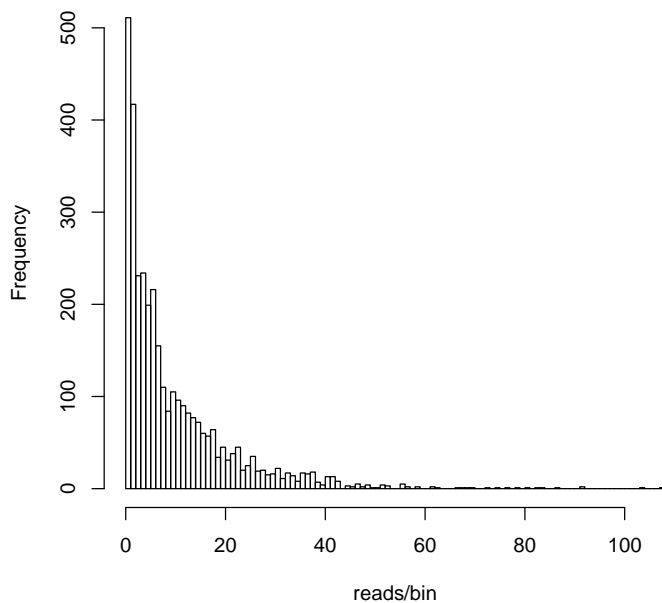
```
> hist(promoter$coupling, breaks=100, main="Promoter CpG densities", xlab="CpG coupling fa
```

Promoter CpG densities

The histogram shows a bimodal distribution of promoter CpG densities. For a histogram of mean rpm values type
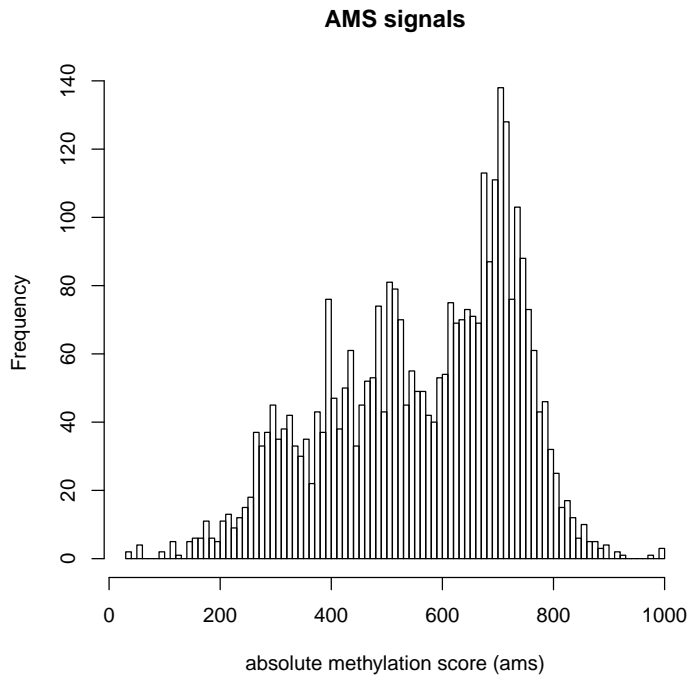
```
> hist(promoter$rpm_A[promoter$rpm_A!=0], breaks=100, main="RPM signals", xlab="reads/bin"
```

RPM signals

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. The *rms* or *ams* values indicate a bimodal distribution of promoter methylation:

```
> hist(promoter$ams_A[promoter$ams_A!=0], breaks=100, main="AMS signals", xlab="absolute m
```

AMS signals

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 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=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.chr22.meth.txt", sep="\t", quote=F, co
```

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

27

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

```
> frames.frame500.step250=read.table(file="frames.chr22.meth.txt", header=T)
```

3.11 Additional MEDIPS SET Objects

In order to identify differentially methylated regions (DMRs) between two different conditions, a second MEDIPS SET has to be created and processed. In our example, the file `MeDIP_DE_chr22.txt` contains MeDIP-seq data of chromosome 22 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:

```
> file=system.file("extdata", "MeDIP_DE_chr22.txt", package="MEDIPS")
> TREAT.SET=MEDIPS.readAlignedSequences(BSgenome="BSgenome.Hsapiens.UCSC.hg19", file=file,

> 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. Here, we will not perform the quality controls or further diagnostic analyses for the TREAT.SET. However, it is recommended to calculate these quality metrics for each data set.

For identification of differentially methylated regions, 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 given in the example file `Input_StemCells_chr22.txt`:

```
> file=system.file("extdata", "Input_StemCells_chr22.txt", package="MEDIPS")
> INPUT.SET=MEDIPS.readAlignedSequences(BSgenome="BSgenome.Hsapiens.UCSC.hg19", file=file,

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

In case of Input data, we do not correct for CpG densties because no enrichment for methylated CpG's was performed. Only the rpm values will be considered. Therefore, the INPUT SET is already sufficiently processed for beeing integrated into the DMR identification process. In the context of this manual, we do not perform the quality controls and further diagnostic analyses for the INPUT.SET. However, we strongly recommend to calculate the quality control metrics as described for the CONTROL.SET, in order to work out the differences between MeDIP and Input data (e.g. saturation-, coverage-, CpG enrichment analyses, and creation of the calibration plot). These analyses can be performed for the INPUT.SET by typing e.g.:

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

> sr.input=MEDIPS.saturationAnalysis(data=INPUT.SET, bin_size=50, extend=400, no_iteration
> MEDIPS.plotSaturation(sr.input)
```

The saturation analysis will show that the reproducibility increases more slowly for Input data 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.

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

> cr.input=MEDIPS.coverageAnalysis(data=INPUT.SET, extend=400, no_iterations=10)
> MEDIPS.plotCoverage(cr.input)

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

The CpG enrichment values indicate that Input data is not as strong enriched for CpGs as MeDIP data.

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

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

> png("CalibrationPlotINPUT.png")
> MEDIPS.plotCalibrationPlot(data=INPUT.SET, linearFit=F, plot_chr="chr22")
> dev.off()
```

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 profiles of the two MEDIPS SETs with respect to the INPUT SET. 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 adjacent 500bp windows:

```
> diff.meth=MEDIPS.methylProfiling(data1=CONTROL.SET, data2=TREAT.SET, input=INPUT.SET, fr
```

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.

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 $ROI = ROI_1, ..., ROI_i, ...ROI_n$, and n is the number of ROIs to be tested. Let the ROIs be of length $m_1, ..., m_n$. 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 $bin_{i,1}, ..., bin_{i,j}, ..., bin_{i,k_i} \epsilon ROI_i$ and $k_i = floor(\frac{m_i}{b})$. 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 rpm (or rms, respectively, see parameter `select`) values of the genomic bins $bin_{i,1}, ..., bin_{i,j}, ..., bin_{i,k_i} \epsilon ROI_i$ of the Control sample significantly differ from the rpm (or rms, respectively, see parameter `select`) 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, in case the $trasnf$ parameter was set to TRUE, the returned mean rms and ams values within the $diff.meth$ object are in log2 scale and were transformed into the consistent interval [0:1000]. However, ratios and p.values were calculated before the data was log2 scaled and shifted into the interval [0:1000]. Ratios and p.values can be calculated based on the rpm or rms values by specifying the parameter `select` of the `MEDIPS.methylProfiling()` function.

You may want to store the received result 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 result 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 result 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 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 perfomed 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` functions has to be $<=$ `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,

[1] Processing input distribution...
[1] Done.
[1] Total number of frames: 102610
[1] Number of frames where control or treatment !=0:  66656
[1] Remaining number of frames with p.value<=1e-04: 7567
[1] Remaining number of frames where control/treatment ratio >= 2: 4108
[1] Estimated rpm threshold for input quantile 0.99 is: 27.4977604916713
[1] Remaining number of frames with control rpm >=27.4977604916713: 433
[1] Remaining number of frames with control/input ratio>=2: 294
[1] [Note: There are 0 frames associated to a p-value==0.]
[1] [Note: There are 0 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 294 frames of length 500bp that remain after the `MEDIPS.selectSignificants()` step. In case, the `MEDIPS.methylProfiling()` function was executed in order to test overlapping frames (i.e. specifying the *step* parameter), overlapping significant frames may be returned to the `diff.meth.control` results table. For these cases it is worthwhile to merge overlapping regions by typing:

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

Please note, merged 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 result 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 chromosome 22) will probably end up in a different number of significant DMRs: First, the calibration parameters were calculated for only one chromosome. 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 one chromosome.

You can write out the obtained regions by typing some suitable R code like:

```
> write.table(diff.meth.control, file = "DiffMethyl.Up.hESCs.bed", sep = "\t", quote = 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 annotate the DMRs. We fall back on the example ROI file `hg19.chr22.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

```
> file=system.file("extdata", "hg19.chr22.txt", package="MEDIPS")
> diff.meth.control.annotated=MEDIPS.annotate(diff.meth.control, anno=file)
```

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 matched 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 separated 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.annotated[,4]))

[1] 25
```

In our example, the 294 identified DMRs on chromosome 22 can be associated to promoter regions of 25 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, do

[1] Processing input distribution...
[1] Done.
[1] Total number of frames: 102610
[1] Number of frames where control or treatment !=0:  66656
[1] Remaining number of frames with p.value<=1e-04: 7567
[1] Remaining number of frames where treatment/control ratio <= 0.5: 1605
[1] Estimated rpm threshold for input quantile 0.99 is: 27.4977604916713
[1] Remaining number of frames with treatment rpm >=27.4977604916713: 64
[1] Remaining number of frames with treatment vs. input ratio>=2: 47
[1] [Note: There are 0 frames associated to a p-value==0.]
[1] [Note: There are 0 frames, where control/treatment ratio = 0 (i.e. control=0).]

> write.table(diff.meth.treat, file = "DiffMethyl.Up.DE.bed", sep = "\t", quote = F, row.n

> file=system.file("extdata", "hg19.chr22.txt", package="MEDIPS")
> diff.meth.treat.annotated=MEDIPS.annotate(diff.meth.treat, anno=file)

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

[1] 28
```

In our example, there are 47 non-overlapping DMRs on chromosome 22 associated to promoter regions of 28 unique transcripts. These DMRs can be interpreted as regions where de-novo 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 adressed in the future. 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 calculation of mean methylation values or identification of DMRs at genome wide short windows remains a bottleneck.

- The `MEDIPS.methylProfiling()` function is a novel approach for the identification of DMRs, 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, identification of DMRs is very static. The definition of fixed windows, although when overlapping windows are allowed, is not very flexible. A dynamic method for the identification of optimized DMRs might be of interest.

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, Hans Lehrach, Ralf Herwig, and James Adjaye. Computational analysis of genome-wide dna methylation during the differentiation of human embryonic stem cells along the endodermal lineage. *Genome Res*, Aug 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.