MEDIPS: genome-wide differential coverage analysis of sequencing data derived from DNA enrichment experiments.

Lukas Chavez*

May 4, 2016

Contents

1 Introduction

MEDIPS was developed for analyzing data derived from methylated DNA immunoprecipitation (MeDIP) experiments [**?**] followed by sequencing (MeDIP-seq) [**?**]. However, *MEDIPS* provides several functionalities for the analysis of other kinds of quantitative sequencing data (e.g. ChIP-seq, MBD-seq, CMS-seq, and others) including calculation of differential coverage between groups of samples as well as saturation and correlation analyses [**?**].

In detail, *MEDIPS* adresses the following aspects in the context of quantitative sequencing data analysis:

- calculating genome wide signal densities at a user specified resolution (counts, rpkm),
- calculating differential coverage comparing two groups of samples,
- estimating the reproducibility for obtaining full genome short read coverage profiles (saturation analysis),
- correcting for copy number variations present in the genomic background of the samples based on Input samples (if available),
- export of raw and normalized data as Wiggle files for visualization in common genome browsers (e.g. the UCSC genome browser).

In addition, *MEDIPS* provides the following MeDIP/MBD-seq specific functionalities:

- analyzing the coverage of genome wide DNA sequence patterns (e.g. CpGs) by the given short reads (or their mapping results, respectively),
- calculating a CpG enrichment factor as a quality control for MeDIP/MBD specific immunoprecipitation,
- calculating genome wide sequence pattern densities (e.g. CpGs) at a user specified resolution and export as wiggle track,
- 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/MBD signals,
- normalization of MeDIP-seq data with respect to local sequence pattern (e.g. CpG) densities (rms, relative methylation score) [**?**].

MEDIPS starts where the mapping tools stop (BAM or BED files) and can be used for any genome of interest. In case a genome of interest is not available as a *BSgenome* package but the sequence of the genome is available, a custom *BSgenome* package can be generated, please see the "How to forge a *BSgenome* data package" manual of the *BSgenome* package.

2 Installation

To install the *MEDIPS* package into your *R* environment, start *R* and enter:

*l.chavez@dkfz.de

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

Next, it is necessary to have a genome of interest available in your *R* environment. Please load the *BSgenome* package

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

and check the available genomes

```
> available.genomes()
```

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 reference genome.

Finally, the *MEDIPS* workflow described below requires access to example data available in the *MEDIPSData* package which can be installed by typing:

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

3 Preparations

First, the *MEDIPS* package has to 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 hg19 library:

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

MEDIPS requires mapping results in BAM or tab-separated (|) BED text files (chr | start | end | name | score | strand) as input. *MEDIPS* can also import precomputed genome coverage from fixedStep wiggle files. In the latter case, all specified chromosomes have to be described completely.

In order to present a typical *MEDIPS* workflow, we access BAM files as well as preprocessed data included in the data package *MEDIPSData*. In order to load the library, please type:

```
> library("MEDIPSData")
```

The example data has been generated based on the following BAM files:

`hESCs.MeDIP.Rep1.chr22.bam` (7.6M), `hESCs.MeDIP.Rep2.chr22.bam` (14M), `hESCs.MeDIP.Rep3.chr22.bam` (9.1M), `hESCs.Input.chr22.bam` (4.9M), `DE.MeDIP.Rep1.chr22.bam` (12M), `DE.MeDIP.Rep2.chr22.bam` (14M), `DE.MeDIP.Rep3.chr22.bam` (13M), and `DE.Input.chr22.bam` (11M)

These BAM files are the *bowtie* [**?**] mapping results of MeDIP-seq data derived from three replicates of human embryonic stem cells (hESCs) and three replicates of differentiated hESCs (definitive endoderm, DE) [**?**]. For each condition there is one BAM file containg the mapping results of corresponding Input-seq data.

In this manual, we access some BAM files located in the `extdata` subdirectory of the *MEDIPSData* package. We have to point to the example BAM files as follows:

```
> bam.file.hESCs.Rep1.MeDIP = system.file("extdata", "hESCs.MeDIP.Rep1.chr22.bam",
+     package = "MEDIPSData")
> bam.file.hESCs.Input = system.file("extdata", "hESCs.Input.chr22.bam",
+     package = "MEDIPSData")
> bam.file.DE.Input = system.file("extdata", "DE.Input.chr22.bam", package = "MEDIPSData")
```

Some other BAM files have been already preprocessed and will be accessed as internally saved RData objects (see below).

Typically, you will directly specify your data file at the `file` parameter. It is also possible to specify the full path to your input file together with the file name.

Next, we define several parameters which will be used throughout the manual. The reference genome is hg19:

```
> BSgenome="BSgenome.Hsapiens.UCSC.hg19"
```

To avoid artefacts caused by PCR over amplification *MEDIPS* determines a maximal allowed number of stacked reads per genomic position by a poisson distribution of stacked reads genome wide and by a given p-value:

```
> uniq=1e-3
```

The smaller the p-value, the more reads at the same genomic position are potentially allowed. Alternatively, all reads mapping to exactly the same genomic position can be maintained (uniq = 0) or replaced by only one representative (uniq = 1).

All reads will be extended to a length of 300nt according to the given strand information:

```
> extend=300
```

As an alternative to the extend parameter, the `shift` parameter can be used. Here, the reads are not extended but shifted by the specified number of nucleotides with respect to the given strand infomation. One of the two parameters `extend` or `shift` has to be 0.

```
> shift=0
```

The genome will be divided into adjacent windows of length 100nt and all further calculations (short read coverage, differential coverage between conditions etc.) will be applied to these windows.

```
> ws=100
```

comment: There is no general recommendation for the window size, because different DNA enrichment experiments can have different resolutions. Smaller window sizes may lose some statistical power due to smaller sequencing counts per window. The choice of an appropriate window size will depend on the available sequencing depth, the expected resolution of the experiment, and on the memory and runtime requirements (the smaller the window size, the higher the memory requirements and the bigger the result table).

In this manual, we are going to process only one chromosome (i.e. chr22). Therefore, we specify the `chr.select` parameter. Please note, the example BAM files contain only data for chr22 anyway.

```
> chr.select="chr22"
```

comment: Make sure that the sequencing data has been mapped to the same version of the reference genome as the BSgenome reference and that the mapping reference has the same chromosome names as the BSgenome reference. By default, MEDIPS will calculate coverage at all chromosomes of the specified BSgenome reference. To avoid unusual chromosmes in the results, please make use of the `chr.select` parameter.

4 Case study: Genome wide methylation and differential coverage between two conditions

Here, we calculate genome wide short read coverage and methylation profiles for the three hESCs and for the three DE replicates [**?**]. In addition, we calculate differential coverage between conditions (here, this will be interpreted as differential methylation).

4.1 Data Import and Preprocessing

First, we create a MEDIPS SET for the first replicate from the undifferentiated hESCs:

```
> hESCs_MeDIP = MEDIPS.createSet(file = bam.file.hESCs.Rep1.MeDIP, BSgenome = BSgenome,
+     extend = extend, shift = shift, uniq = uniq, window_size = ws, chr.select = chr.select)
```

where the parameters remain as defined in section Preparations. Subsequently, we have to concatenate the remaining two replicates to the first MEDIPS SET resulting in a list of MEDIPS SETs. In principle, this would look like:

```
> bam.file.hESCs.Rep2.MeDIP = system.file("extdata", "hESCs.MeDIP.Rep2.chr22.bam",
+     package = "MEDIPSData")
> hESCs_MeDIP = c(hESCs_MeDIP, MEDIPS.createSet(file = bam.file.hESCs.Rep2.MeDIP,
+     BSgenome = BSgenome, extend = extend, shift = shift, uniq = uniq,
+     window_size = ws, chr.select = chr.select))
```

comment: Concatenating MEDIPS SETs into a list of MEDIPS SETs does not merge these MEDIPS SETs. All samples will be treated as replicates.

However, here we load the preprocessed lists of MeDIP-seq MEDIPS SETs available in the *MEDIPSData* package:

```
> data(hESCs_MeDIP)
> data(DE_MeDIP)
```

For the Input-seq data sets, we explicitely create MEDIPS SETs (also called INPUT SETs) as follows

```
> hESCs_Input = MEDIPS.createSet(file = bam.file.hESCs.Input, BSgenome = BSgenome,
+     extend = extend, shift = shift, uniq = uniq, window_size = ws, chr.select = chr.select)
> DE_Input = MEDIPS.createSet(file = bam.file.DE.Input, BSgenome = BSgenome,
+     extend = extend, shift = shift, uniq = uniq, window_size = ws, chr.select = chr.select)
```

comment: Please consider the parameter paired of the MEDIPS.createSet function for importing paired-end sequening data.

It is possible to add as many samples to each list as available. This is valid for the MeDIP and Input sets.

For CpG density dependent normalization of MeDIP-seq data, we need to generate a coupling set. The coupling set must be created based on the same reference genome, the same set of chromosomes, and with the same window size used for the MEDIPS SETs. For this, we specify the first MEDIPS SET in the hESCs object as reference, but any of the other MEDIPS SETs would be fine as well, because all of them consist of the same set of chromosomes (here chr22 only, hg19) and have been generated with the same window size.

```
> CS = MEDIPS.couplingVector(pattern = "CG", refObj = hESCs_MeDIP[[1]])
```

4.2 Coverage, methylation profiles and differential coverage

It is possible to calculate genome wide coverage and methylation profiles for only one MEDIPS SET or for only one group of MEDIPS SETs using the function MEDIPS.meth. However, in this case study we also want to calculate differential coverage (i.e. differential methylation) between two conditions. Whenever two groups of MEDIPS SETs are provided to the MEDIPS.meth function differential coverage will be calculated.

```
> mr.edgeR = MEDIPS.meth(MSet1 = DE_MeDIP, MSet2 = hESCs_MeDIP, CSet = CS,
+     ISet1 = DE_Input, ISet2 = hESCs_Input, p.adj = "bonferroni", diff.method = "edgeR",
+     MeDIP = T, CNV = F, minRowSum = 10)
```

Parameters that can be specified are:

- MSet1: first group of MEDIPS SETs. Please specify at least one set.
- MSet2: second group of MEDIPS SETs. Differential coverage will be calculated, if MSet1 and MSet2 are not empty.
- CSet: a coupling set (typically a CpG coupling set). Mandatory for normalization of MeDIP data. Can be skipped, if MeDIP=F (see below).
- ISet1: first group of INPUT SETs (corresponding to MSet1).
- ISet2: second group of INPUT SETs (corresponding to MSet2). Copy number variations will be calculated, if ISet1 and ISet2 are not empty. Can be swithced off by setting CNV=F (see below).
- p.adj: in order to correct p.values for multiple testing, *MEDIPS* uses *R*'s p.adjust function. Therefore, the following methods are available: holm, hochberg, hommel, bonferroni, BH, BY, fdr, none.
- diff.method: method for calculating differential coverage. Available methods: ttest and edgeR. The ttest method will be calculated only in case there are at least three replicates/MEDIPS SETs per group. The ttest method can be applied to the rpkm or CpG density normalized rms values as specified by the type parameter

(see below). The `ttest` method adds four vectors to the result table: `score.log2.ratio`, `score.p.value`, `score.adj.p.value`, and `score` where `score = (-log10(p.value)*10)*log(ratio)`. As an alternative, *edgeR* can be applied for testing differential coverage at genome wide windows even for less than 3 replicates per group. However, in case there is only one MEDIPS SET per group, the dispersion will be set to bcv^2 where $bcv = 0.01$ (please consider section "What to do if you have no replicates" of *edgeR*'s User's Guide). When applying *edgeR*, the weighted trimmed mean of M-values (TMM) method is used to calculate scale factors between libraries. The *edgeR* method will be applied to the counts of the genome wide windows. The *edgeR* method adds four vectors to the result table which are *edgeR*'s exactTest standard output: `edgeR.logFC`, `edgeR.logCPM`, `edgeR.p.value`, and `edgeR.adj.p.value`.

- `MeDIP`: This parameter determines, if a CpG density dependent relative methylation scores (rms) will be calculated for the MEDIPS SETs given at the slots `MSet1` and `MSet2`.
- `CNV`: In case there are INPUT SETs provided at both Input slots (i.e. `ISet1` and `ISet2`), copy number variation will be tested by applying the package `DNAcopy` to the window-wise log-ratios calculated based on the the means per group. By setting `CNV`=F this function will be disabled. Please note, the function `MEDIPS.addCNV` allows to run the CNV analysis on two groups of INPUT SETs using another (typically increased) window size. Subsequently, the `MEDIPS.addCNV` function matches its CNV results to a given result table previously generated by the `MEDIPS.meth` function. This allows for calculating CNVs for larger windows what might be more appropriate in case of low Input sequencing depth.
- `type`: In case `diff.method` has been set to `ttest`, this parameter specifies, if differential coverage is calculated based on the rpkm or rms values. This parameter is ignored in case the *edgeR* method. *edgeR* will always be applied to the count data.
- `chr`: to process only a selected set of chromosomes, e.g. c("chr1", "chr2")).
- `minRowSum`: threshold for a minimum sum of counts across all samples per window (default=10). Windows with lower coverage will not be tested for differential coverage.
- `diffnorm`: To normalize for different library sizes and/or for different enrichment efficiencies, normalization can be enabled by setting this parameter to tmm, quantile, rpkm or rms (or none). If differential enrichment between conditions is calcualted based on edgeR (diff.method=edgeR), tmm (default) or quantile (or none) are possible, while rpkm and rms are only available when diff.method=ttest. It has been proposed [**?**] that quantile normalization can correct for varying DNA enrichment efficiencies. When enabled, quantile normalization will be applied to the count table (including all genomic windows and all samples) prior to testing for differential coverage between conditions. *warning: In case of quantile normalisation, the counts - but not rpkm and rms values - of the returned result table will be quantile normalized.*

The returned object (here `mr.edgeR`) is a list of vectors (all of the same length) where the rows correspond to genome wide windows (i.e. the result table). The number of vectors (or columns of the result table) depends on the number of provided MEDIPS SETs, the number of INPUT SETs, and on the specification of several parameters including `diff.methods`, `MeDIP`, and `CNV`. For each window and for each MEDIPS or INPUT SET, there will be columns for the `counts` and `rpkm` values. For MEDIPS SETs there can also be columns for `rms` values (if `MeDIP`=T). For each condition with more than one provided MEDIPS or INPUT SET, respectively, there will be an additional column for the mean over the counts and rpkm values (and for rms, if available). Moreover, there are columns for log2 ratios (MSet1/MSet2), p.values, and adjusted p.values when differential coverage has been calculated. Optionally, there will be a vector of log2 ratios (ISet1/ISet2) as a result of the CNV analysis, in case two groups of INPUT SETs have been provided (and in case `CNV`=T) or in case a CNV analysis has been added afterwards by applying the function `MEDIPS.addCNV` (see below).

4.3 Differential coverage: selecting significant windows

As we have processed two groups of MEDIPS SETs, we are now interested in genomic windows which show significant differential coverage:

```
> mr.edgeR.s = MEDIPS.selectSig(results = mr.edgeR, p.value = 0.1, adj = T,
+     ratio = NULL, bg.counts = NULL, CNV = F)

Total number of windows: 513046
Number of windows tested for differential methylation: 138262
Remaining number of windows with adjusted p.value<=0.1: 64
```

Here, we set a thresold of 0.1 for the adjusted p.values in order to select significant windows. There are the following filter criteria available:

- p.value: this is the p.value threshold as calculated either by the ttest or *edgeR* method
- adj: this parameter specifies whether the p.value or the adjusted p.values is considered
- ratio: this parameter sets an additional thresold for the ratio where the ratio is either score.log2.ratio or edgeR.logFC depending on the previously selected method. Please note, the specified value will be transformed into log2 internally.
- bg.counts: as an additional filter parameter, it is possible to require a minimal number of reads per window in at least one of the MEDIPS SET groups. To apply this condition, the mean of the counts per group is considered. The parameter bg.counts can either be a concrete integer or an appropriate column name of the result matrix. By specifying a column name (here e.g. bg.counts="hESC.Input.bam.counts" or bg.counts="MSets1.counts.mean"), the 0.95 quantile of the according genome wide count distribution is determined and used as a minimal background threshold (please note, only count columns are reasonable).
- CNV: The information on CNVs present in the samples of interest can be used for correcting differential coverage observed in the corresponding IP data (e.g. MeDIP or ChIP data). In the given example data, we do not expect extensive events of copy number variation between pluripotent and differentiated hESCs because both samples have been derived from the same cell line. However, samples derived from e.g. cancer tissues might show an relevant occurence of copy number variations. Therefore, when comparing MeDIP- or ChIP-seq data derived from cancer tissue against MeDIP- or ChIP-seq data derived from corresponding healthy tissue, an event of differential coverage might be explained in part or entirely by a local CNV. In case Input data has been provided for both conditions, *MEDIPS* is capable of calculating genome wide CNV ratios by employing the package DNAcopy. In case the parameter CNV is set to TRUE, *MEDIPS* will consider only genomic windows having a CNV corrected IP ratio higher than the specified ratio threshold (specification of the ratio parameter is required in this case).

4.4 Merging neighboring significant windows

After having identified and extracted genomic windows showing significant differential coverage between conditions, it remains of interest to merge neighboring significant windows into a larger continuous region. Please note, the result object mr.edgeR.s returned by the MEDIPS.selectSig contains genomic regions that show differential coverage into both directions, i.e. either a higher coverage in the group of samples in MSet1 (here DE) over the group of samples in MSet2 (here hESCs) or vice versa. In order to isolate the subset of genomic regions that show e.g. higher coverage in MSet1 over MSet2, we use the following *R* syntax:

```
> mr.edgeR.s.gain = mr.edgeR.s[which(mr.edgeR.s[, grep("logFC", colnames(mr.edgeR.s))] >
+     0), ]
```

Subsequently, the function MEDIPS.mergeFrames can be applied to merge all adjacent significant regions into one region which will be finally regarded as one event of differential coverage:

```
> mr.edgeR.s.gain.m = MEDIPS.mergeFrames(frames = mr.edgeR.s.gain, distance = 1)
```

Here we interpret the selected and merged genomic regions as differentially methylated regions (DMRs) showing gain of methylation during differentiation.

The distance parameter allows for merging windows having an according gap in between. Please note, merged windows are represented only by their genomic coordinates and do not contain any summarized values (counts, rpkm, ratios etc.).

4.5 Extracting data at regions of interest

MEDIPS provides the functionality to select subsets of the result matrix returned by the MEDIPS.meth function according to any given set of regions of interest (ROIs). As an example, we consider the set of merged windows (here mr.edgeR.s.gain.m, see previous section) as a set of ROIs:

```
> columns = names(mr.edgeR)[grep("counts", names(mr.edgeR))]
> rois = MEDIPS.selectROIs(results = mr.edgeR, rois = mr.edgeR.s.gain.m,
+     columns = columns, summarize = NULL)
```

The function MEDIPS.selectROIs will select all genomic windows in the original result table (here mr.edgeR) which are included in the boundaries defined by the genomic coordinates of the given set of ROIs. The number of returned windows depends on the window size and on the length of the provided ROIs. Please note, only selected columns will

be returned as determined by the columns parameter. In the given example, all columns that contain count values will be extracted.

As an alternative, it is also possible to calculate mean values over the extracted windows for each ROI, a behaviour that is controlled by the parameter summarize:

```
> rois.s = MEDIPS.selectROIs(results = mr.edgeR, rois = mr.edgeR.s.gain.m,
+     columns = columns, summarize = "avg")
```

For each ROI and for each specified column, there will be one mean value returned.

5 Quality controls

MEDIPS provides three different quality controls. In this section, the quality controls are demonstrated for the MeDIP-seq hESCs sample hESCs_Rep1_MeDIP.bam.

5.1 Saturation analysis

The saturation analysis addresses the question, whether the given set of mapped reads is sufficient to generate a saturated and reproducible coverage profile of the reference genome. Only if there is a sufficient number of short reads, the resulting genome wide coverage profile will be reproducible by another independent set of a similar number of short reads. The saturation analysis is not specific for MeDIP-seq data and can be applied to other types of sequencing data like e.g. ChIP-seq.

```
> sr = MEDIPS.saturation(file = bam.file.hESCs.Rep1.MeDIP, BSgenome = BSgenome,
+     uniq = uniq, extend = extend, shift = shift, window_size = ws, chr.select = chr.select,
+     nit = 10, nrit = 1, empty_bins = TRUE, rank = FALSE)
```

The saturation analysis divides the total set of available regions into two distinct random sets (A and B) of equal size. Both sets A and B are again divided into random subsets of equal size where the number of subsets is determined by the parameter nit (default=10). For each set, A and B, the saturation analysis iteratively selects an increasing number of subsets and calculates short read coverage at genome wide windows where the window sizes are defined by the window_size parameter. In each iteration step, the resulting genome wide coverages for the current subsets of A and B are compared using pearson correlation. As the number of considered reads increases during each iteration step, it is assumed that the resulting genome wide coverages become more similar, a dependency that is expressed by an increased correlation.

It has to be noted that the saturation analysis can be performed on two independent sets of short reads only. Therefore, a true saturation for one given sample can only be calculated for half of the available short reads. As it is of interest to examine the reproducibility for the total set of available short reads of the given sample, the saturation analysis is followed by an estimated saturation analysis. For the estimated saturation analysis, the full set of given regions is artifically doubled by considering each given region twice. Subsequently, the described saturation analysis is performed on the artificially doubled set of regions (see also Supplementary Methods in [**?**]).

The results of the saturation and of the estimated saturation analysis can be viewed by typing

```
> sr
```

```
$distinctSets
       [,1]      [,2]
 [1,]     0 0.0000000
 [2,]  7539 0.1469049
 [3,] 15078 0.2516678
 [4,] 22617 0.3313464
 [5,] 30156 0.3999554
 [6,] 37695 0.4519189
 [7,] 45234 0.4993105
 [8,] 52773 0.5372634
 [9,] 60312 0.5667903
[10,] 67851 0.5960764
```

```
[11,] 75396 0.6244021

$estimation
        [,1]      [,2]
 [1,]      0 0.0000000
 [2,]   7539 0.1562992
 [3,]  15078 0.2754212
 [4,]  22617 0.3665482
 [5,]  30156 0.4312434
 [6,]  37695 0.4824616
 [7,]  45234 0.5283587
 [8,]  52773 0.5662905
 [9,]  60312 0.5995332
[10,]  67851 0.6269652
[11,]  75390 0.6503878
[12,]  82929 0.6723364
[13,]  90468 0.6912637
[14,]  98007 0.7089813
[15,] 105546 0.7248347
[16,] 113085 0.7391568
[17,] 120624 0.7519015
[18,] 128163 0.7638021
[19,] 135702 0.7736465
[20,] 143241 0.7826988
[21,] 150793 0.7912300

$numberReads
[1] 150793

$maxEstCor
[1] 1.50793e+05 7.91230e-01

$maxTruCor
[1] 7.539600e+04 6.244021e-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 (here `sr`, first column: total number of considered reads, second column: 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).

In addition, the results can be visualized as shown in Figure **??** by typing

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

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

- `nrit`: methods that randomly select data entries may be processed several times in order to obtain more stable results. By specifying the `nrit` 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 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 wide coverages. If there occur genomic bins which contain no overlapping regions, neither from the subset(s) of `A` nor from the subset(s) of `B`, these windows 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 (i.e. spearman correlation). Setting this parameter to TRUE is a more robust approach which reduces the effect of possible occuring outliers (i.e. windows with very high counts) on the correlation.

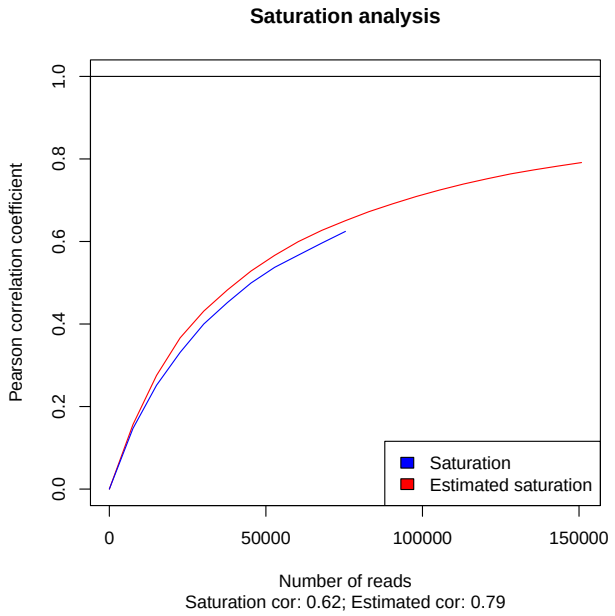

Figure 1: **Saturation analysis.** The saturation analysis indicates the reproducibility of the genome wide coverage at regular genomic intervals given an increasing sequencing depth.

5.2 Correlation between samples

Genome wide short read coverage profiles are expected to be similar for biological or technical replicates given a sufficient sequencing depth (compare section `Saturation Analysis`). The `MEDIPS.correlation` function comapares genome wide coverage profiles of given MEDIPS SETs and returns a correlation matrix containing pair-wise correlations.

```
> cor.matrix = MEDIPS.correlation(MSets = c(hESCs_MeDIP, DE_MeDIP, hESCs_Input,
+     DE_Input), plot = T, method = "pearson")
```

5.3 Sequence Pattern Coverage

The main idea of the sequence pattern coverage analysis is to test the number of CpGs (or any other predefined sequence pattern) covered by the given short reads. In addition, the depth of coverage per CpG is tested. The sequence pattern coverage analysis can be started by typing

```
> cr = MEDIPS.seqCoverage(file = bam.file.hESCs.Rep1.MeDIP, pattern = "CG",
+     BSgenome = BSgenome, chr.select = chr.select, extend = extend, shift = shift,
+     uniq = uniq)
```

The sequence pattern coverage results object (here `cr`) contains four different slots:

- `cov.res`: is a vector of length equal to the total number of sequence patterns (e.g. CpGs). The entries correspond to the number of overlapping regions (coverage).
- `pattern`: the tested sequence pattern (e.g. CpG)
- `numberReads`: the total number of tested reads
- `numberReadsWO`: the number of reads which do not cover a tested sequence pattern

The results of the coverage analysis can be visualized by a pie chart as shown in Figure **??**:

```
> MEDIPS.plotSeqCoverage(seqCoverageObj=cr, type="pie", cov.level = c(0,1, 2, 3, 4, 5))
```

```
Creating summary...
```

The visualized coverage levels can be adjusted by the `cov.level` parameter.

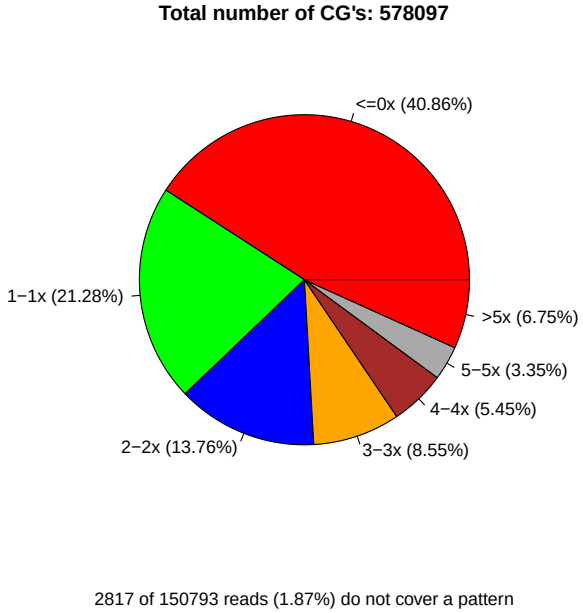

Figure 2: **Coverage analysis- pie chart** The coverage analysis illustrates the fraction of CpGs covered by the given reads according to their coverage level.

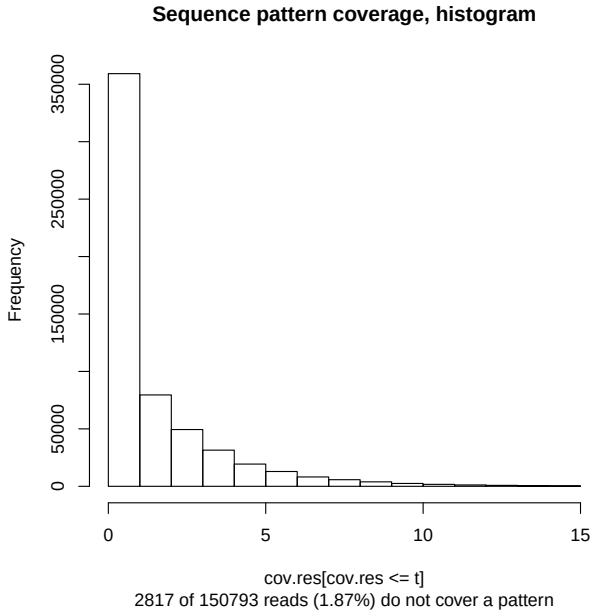

Figure 3: **Coverage analysis- histogram** The histogram shows the distribution of CpG coverages. The parameter `t` specifies the maximal coverage depth to be plotted (here: 15 reads).

Alternatively, a histogram can be plotted to visalizes the total number of CpGs that have been covered at different level (see Figure **??**):

```
> MEDIPS.plotSeqCoverage(seqCoverageObj=cr, type="hist", t = 15, main="Sequence pattern coverage, histo
```

5.4 CpG Enrichment

As a quality check for the enrichment of CpG rich DNA fragments obtained by the immunoprecipitation step of a MeDIP/MBD experiment, *MEDIPS* provides the functionality to calculate CpG enrichment values. The main idea is

to test the enrichment of CpGs within the genomic regions covered by the given set of short reads compared to the full 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.

```
> er = MEDIPS.CpGenrich(file = bam.file.hESCs.Rep1.MeDIP, BSgenome = BSgenome,
+     chr.select = chr.select, extend = extend, shift = shift, uniq = uniq)
```

The enrichment results object (here `er`) contains several slots which 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 (i.e. sequencing of none-enriched DNA fragments), the enrichment values should be close to 1. In contrast, a MeDIP-seq experiment should return CpG rich sequences what will be indicated by increased CpG enrichment scores.

6 Miscellaneous

In this section we discuss several additional functionalities currently available in the *MEDIPS* package.

6.1 Processing regions of interest

Instead of caluclating coverage and differential coverage at genome wide small windows, it is also possible to perform targetd analyses of regions of interest (ROI's, e.g. exons, promoter regions, CpG islands etc.). Here, the alignment files have to be imported using the function `MEDIPS.createROIset`. The function `MEDIPS.createROIset` has the same parameters as `MEDIPS.createSet` except that the `window_size` parameter has been replaced by the `ROI` parameter. The `ROI` parameter expects a data.frame with columns "chr", "start", "end" and "name" defining the regions of interest. Furthermore, ROIs can be divided into bins prior to calculating sequencing coverage, a behaviour controlled by the parameter `bn`. *comment: Please note, all ROIs will be divided into the same number of bins regardless of their length, possibly resulting into bins of different sizes.*

6.2 Export Wiggle Files

MEDIPS allows to export genome wide coverage profiles as wiggle files for visualization in common genome browsers.

```
> MEDIPS.exportWIG(Set = hESCs_MeDIP[[1]], file = "hESC.MeDIP.rep1.wig",
+     format = "rpkm", descr = "")
```

- `Set`: a MEDIPS SET to be exported as a wiggle file. Required when the parameter `format` is `count`, `rpkm`, or `rms`.
- `CSet`: a COUPLING SET to be exported as a wiggle file. Required when the parameter `format` is `pdensity` or `rms`.
- `file`: the output file name
- `format`: can be either count, rpkm, or rms for a MEDIPS SET or pdensity for a COUPLING SET.
- `descr`: a track description for the wiggle file

6.3 Merging MEDIPS SETs

A MEDIPS SET represents the mapping results of a biological sample of interest, typically a biological or technical replicate. However, a biological or technical replicate might be sequenced several times or distributed over different lanes resulting in several bam files. Instead of merging associated bam files in advance of a *MEDIPS* analysis, the *MEDIPS* package allows to merge MEDIPS SETs generated from associated bam files.

```
> Input.merged = MEDIPS.mergeSets(MSet1 = hESCs_Input, MSet2 = DE_Input,
+     name = "Input.hESCs.DE")
```

Please note, MEDIPS SETs to be merged are required to be previously generated based on the same reference genome, chromosome set, and window size. The `extend`, `shift` or `uniq` parameters can be different and are left blank in the resulting MEDIPS SET specification. The parameter `name` assigns a new name to the merged MEDIPS SET. Consequently, the merged MEDIPS SET does not have a concrete path and file association anymore and cannot be used as input for the `MEDIPS.addCNV` function.

6.4 Annotation of significant windows

It is of interest to annotate genomic windows with known genome characteristics like transcription start sites, exons, CpG islands etc. *MEDIPS* provides a basic function for adding additional annotation columns to the result table. For this, an annotation object is required which contains the annotation ids, chromosome names, start and end positions. As an example, *MEDIPS* allows to generate such an annotation object by accessing BioMart [**?**]:

```
> anno.mart.gene = MEDIPS.getAnnotation(dataset = c("hsapiens_gene_ensembl"),
+     annotation = c("GENE"), chr = "chr22")
```

Annotation of the result matrix (either the full matrix or a subset of the matrix as selected by e.g. the `MEDIPS.selectSig` function) can now be performed as follows:

```
> mr.edgeR.s = MEDIPS.setAnnotation(regions = mr.edgeR.s, annotation = anno.mart.gene)
```

An annotation is assigned to a genomic window, if their genomic coordinates overlap.

6.5 addCNV

In case there are INPUT SETs provided at both Input slots (i.e. `ISet1` and `ISet2`) of the `MEDIPS.meth` function, copy number variation will be tested by applying the package `DNAcopy` to window-wise log-ratios calculated based on the the means per group. However, by setting `CNV`=F copy number variation analysis will be disabled avoiding a computational demanding CNV analysis at previously determined (typically short) genome wide windows. The function `MEDIPS.addCNV` allows to run the CNV analysis on two groups of INPUT SETs using another (typically increased) window size. Subsequently, the `MEDIPS.addCNV` function matches its CNV results to the given result table as previously generated by the `MEDIPS.meth` function. This allows for calculating CNVs for larger windows what might be more appropriate in case of low sequencing depth of Input-seq data.

```
> mr.edgeR = MEDIPS.addCNV(cnv.Frame = 10000, ISet1 = hESCs_Input, ISet2 = DE_Input,
+     results = mr.edgeR)
```

6.6 Calibration Plot

MeDIP-seq data is transformed into genome wide relative methylation scores by a previously presented CpG dependent normalization method [**?**]. Normalization is based on the dependency between short read coverage and CpG density at genome wide windows. This dependency has been originally demonstrated by Down et al. [**?**] and can be visualized as a calibration plot.

```
> MEDIPS.plotCalibrationPlot(CSet = CS, main = "Calibration Plot", MSet = hESCs_MeDIP[[1]],
+     plot_chr = "chr22", rpkm = TRUE, xrange = TRUE)
```

- `CS`: the COUPLING SET
- `ISet`: an INPUT SET (i.e. a MEDIPS SET created from Input sequening data). No linear regression will be calculated for the Input Set.
- `main`: the main of the figure
- `MSet`: the MEDIPS SET object
- `plot_chr`: the chromsome for which the calibration plot will be generated
- `rpkm`: specifies, if the MeDIP data range will be plotted in count or rpkm format. It is necessary to set rpkm=T, if both, a MSet and an ISet are given and plotted at the same time.
- `xrange`: if set to TRUE, only the lower data range of the calibration plot will be visualized

Please note, it is strongly recommended to direct the output to a graphics device like e.g. png().

6.7 Comments on the experimental design and Input data

There are two common questions for conducting and analyzing MeDIP or any other kind of enrichment/immunoprecipitation (IP) based sequencing data: first, it is of interest to identify enriched regions in an IP sample in order to reveal e.g. methylated regions, the occurrence of histone modifications, or to localize transcription factor binding sites. Second, it is of interest to identify differential short read coverage comparing two conditions (here treatment vs. control) in order to identify genomic regions that show e.g. differential methylation or differential transcription factor binding. An intutive approach for identifying differential coverage is to first calculate condition-wise enrichments (or 'peaks') followed by mutual subtraction of these 'peak sets'. However, this approach has several limitations (not discussed here) and the *MEDIPS* package follows an alternative strategy directly comparing the groups of IP samples against each other.

For the identification of sample-wise local enrichments, any peak finder appropriate for the analyzed data type may be applied (differences of peak finding tools are not discussed here). In addition, it is recommended to include Input data for estimating a background coverage distribution and local coverage bias. Most peak finders are capable of incorporating Input data. For MeDIP data, a CpG density bias has been observed that is not considered by standard peak callers. MEDIPS allows to incorporate the CpG density into the short-read counts resulting in relative methylation values (rms), and we have shown an improved correlation of rms values to bisulfite sequencing data compared to counts or rpkm values [**?**]. However, the resulting range of rms values depends on several unknown factors including the total abundance of methyalted cytosines in the sample or high outlier signals which might be present even in Input data. In case reference bisulfite data of selected genomic regions is available (low to high methylation), it is possible to associate the range of MeDIP-seq derived rms values to the bisulfite derived data range (i.e. percentage of methylated cytosines per window) [**?**]. However, without reference data, it remains an insufficiently solved problem to estimate thresholds for low, intermediate, and high metylation, or to caluculate significant local enrichments based on CpG density corrected data in order to identify methylated regions (or 'peaks') over background.

For the identification of differential coverage, it can be assumed that any experiment independent bias, like local CpG density, affects the short read coverage in both samples (control and treatment) the same way. Therefore, no normalization of CpG density or other experiment independent factors may be performed when differential coverage between two groups of IP samples is calculated. As a consequence, Input samples for the control and for the treatment samples might not be necessary. However, there can exist differences in the genomic backgrounds of the control and treatment samples caused by e.g. copy number variations in cancer versus healthy cells. In such cases, it is necessary to explicitly create independent Input samples for the control and for the treatment samples in order to reveal local background differences that can influence the identification of differential coverage comparing the according IP samples. *MEDIPS* (1.10.0 or higher) can process an arbitrary number of Input samples per condition, if available, and allows for considering copy number variations observed in the Input data when differential coverage is calculated for the IP data.

In summary, the need for Input data depends on two conditions: the experimental design ("Am I interested in IP enrichments over Input or do I want to calculate differential coverage between groups of IP samples?") and the genomic background or different experimental conditions of the biological samples ("Do my samples have the same genomic background?").