Package 'MEDIPS'

April 5, 2014

Type Package

Title (MeD)IP-seq data analysis

Version 1.12.0

Date 2013-07-09

Author Lukas Chavez, Matthias Lienhard, Joern Dietrich

Maintainer Lukas Chavez <lchavez@liai.org>

Description 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.

License GPL (>=2)

LazyLoad yes

biocViews Sequencing, DNAMethylation, CpGIsland,DifferentialExpression, HighThroughputSequencing, ChIPseq,Preprocessing, QualityControl, Visualization

Depends R (>= 3.0), BSgenome, DNAcopy

Imports Biostrings, BSgenome, Rsamtools, graphics, gtools, IRanges,methods, stats, utils, GenomicRanges, edgeR, GenomicFeatures,DNAcopy, biomaRt, rtracklayer

Suggests BSgenome, BSgenome.Hsapiens.UCSC.hg19, MEDIPSData

R topics documented:

`MEDIPS-package` *(MeD)IP-seq data analysis*

Description

MEDIPS was developed for analyzing data derived from methylated DNA immunoprecipitation (MeDIP) experiments followed by sequencing (MeDIP-seq). Nevertheless, several functionalities may be applied to other types of sequencing data (e.g. differential coverage or testing the saturation of ChIP-seq data). MEDIPS addresses several aspects in the context of MeDIP-seq data analysis including basic data processing, several quality controls, normalization, and identification of differential coverage.

Details

Package:	MEDIPS
Type:	Package
Version:	1.10.0
Date:	2013-02-25
License:	GPL (>=2)
LazyLoad:	yes
Depends:	R (>= 2.12.0), BSgenome, DNAcopy

Author(s)

Lukas Chavez, Matthias Lienhard, Joern Dietrich

Maintainer: Lukas Chavez <lchavez@liai.org>

References

Chavez L, Jozefczuk J, Grimm C, Dietrich J, Timmermann B, Lehrach H, Herwig R, Adjaye J., Computational analysis of genome-wide DNA methylation during the differentiation of human embryonic stem cells along the endodermal lineage, Genome Res. 2010 Oct;20(10):1441-50. Epub 2010 Aug 27.

COUPLINGset-class *COUPLINGset class and internal functions*

Description

COUPLINGset class is used in the MEDIPS library to store and extract information generated during the creation of a coupling vector.

Objects from the Class

Objects of the classes contain information about sequence pattern information, included chromosomes, and further parameter settings. A COUPLING SET object is created by the MEDIPS.couplingVector() function. According slots will be filled during the workflow.

Slots

`genome_name`**:** Object of class `"character"` : the refernce genome

`window_size`**:** Object of class `"numeric"` : the window size for the genome vector

`chr_names`**:** Object of class `"character"` : the names of the chromosomes included within the MEDIPS/COUPLING SET

chr_lengths: Object of class "numeric" : the lengths of the chromosomes included within the MEDIPS/COUPLING SET

seq_pattern: Object of class "character" : the sequence pattern (e.g. CG)

genome_CF: Object of class "numeric" : the coupling factor at the genomic bins

number_pattern: Object of class "numeric" : the total number of sequence pattern

Methods

genome_name signature(object = "COUPLINGset"): extracts the reference genome of the COUPLING SET

window_size signature(object = "COUPLINGset"): extracts the window size from the window_size slot COUPLING SET

chr_names signature(object = "COUPLINGset"): extracts the names of the chromosomes included within the COUPLING SET

chr_lengths signature(object = "COUPLINGset"): extracts the length of the chromosomes included within the COUPLING SET

seq_pattern signature(object = "COUPLINGset"): extracts the sequence pattern (e.g. CpG)

genome_CF signature(object = "COUPLINGset"): extracts the coupling factor at the genomic bins

number_pattern signature(object = "COUPLINGset"): extracts the total number of sequence pattern

show signature(object = "COUPLINGset"): prints a summary of the COUPLING SET object content

Author(s)

Lukas Chavez, Matthias Lienhard, Joern Dietrich

Examples

```
showClass("COUPLINGset")
```

MEDIPS.addCNV *Function to run a copy number variation analysis.*

Description

Function calculates a CNV analysis based on two INPUT SETs by employing the DNAcopy package. The results are attached to a provided result table.

Usage

```
MEDIPS.addCNV(ISet1, ISet2, results, cnv.Frame=1000)
```

Arguments

ISet1	First group of INPUT SETs
ISet2	Second group of INPUT SETs
results	result table as returned by the MEDIPS.meth function
cnv.Frame	window size used for calculating CNVs. Can be of different size than the result table.

Value

The result table with an additional column containing DNAcopy's log-ratio.

Author(s)

Joern Dietrich

Examples

```
library(MEDIPSData)
library("BSgenome.Hsapiens.UCSC.hg19")

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")

hESCs.Input = MEDIPS.createSet(file=bam.file.hESCs.Input, BSgenome="BSgenome.Hsapiens.UCSC.hg19", extend=250, sh
DE.Input = MEDIPS.createSet(file=bam.file.DE.Input, BSgenome="BSgenome.Hsapiens.UCSC.hg19", extend=250, shift=0,

data(resultTable)

resultTable = MEDIPS.addCNV(cnv.Frame=10000, ISet1=hESCs.Input, ISet2=DE.Input, results=resultTable)
```

MEDIPS.annotate *Funtion to annotate given genomic coordinates.*

Description

This function has been deprecated. Please see MEDIPS.getAnnotation and MEDIPS.setAnnotation instead.

Usage

```
MEDIPS.annotate(region, anno)
```

Arguments

region	a matrix that contains row-wise genomic regions, e.g. DMRs. The columns are: chromosome, start, stop.
anno	the annotation data object contains row-wise the genomic coordinates of annotations. The columns are: chromosome, start, stop, ID

Value

The annotation function returns a matrix where the rows contain the regions from the given frames object (here DMRs) and the columns are:

chr	the chromosome name of the DMR
start	the start position of the DMR
stop	the stop position of the DMR
annotation	the name of the annotation

Author(s)

Joern Dietrich, Matthias Lienhard

Examples

```
print("The function has been deprecated. Please see MEDIPS.getAnnotation and MEDIPS.setAnnotation instead.")
```

MEDIPS.correlation *Calculates pairwise Pearson correlations between provided MEDIPS SETs*

Description

The function calculates genome wide Pearson correlations between all pairs of provided MEDIPS SETs.

Usage

```
MEDIPS.correlation(MSets=NULL, plot = T, method="pearson")
```

Arguments

MSets	a concatenated set of MEDIPS SETs
plot	if specified, the correlation will be depicted as a scatter plot
method	default: pearson; alternatives: kendall, spearman

Value

a correlation matrix

Author(s)

Lukas Chavez

Examples

```
library(MEDIPSData)
data(hESCs_MeDIP)
data(DE_MeDIP)

correlation = MEDIPS.correlation(MSets=c(hESCs_MeDIP[[1]], DE_MeDIP[[1]]), plot = FALSE)
```

MEDIPS.couplingVector *Calculates the sequence pattern densities at genome wide windows.*

Description

The function calculates the local densities of a defined sequence pattern (e.g. CpGs) and returns a COUPLING SET object which is necessary for normalizing MeDIP data.

Usage

```
MEDIPS.couplingVector(pattern="CG", refObj=NULL)
```

Arguments

pattern	defines the sequence pattern, e.g. CG for CpGs.
refObj	a MEDIPS Set or MEDIPS ROI Set that serves as reference for the genome and window parameters.

Value

A COUPLING SET object.

Author(s)

Lukas Chavez

Examples

```
library("MEDIPSData")
library("BSgenome.Hsapiens.UCSC.hg19")

data(hESCs_MeDIP)
CS = MEDIPS.couplingVector(pattern="CG", refObj=hESCs_MeDIP)
```

MEDIPS.coverageAnalysis

The function identifies the number of CpGs (or any other predefined sequence pattern) covered by the given short reads.

Description

This function has been deprecated. Please see MEDIPS.seqCoverage instead.

Usage

```
MEDIPS.coverageAnalysis(data=NULL, coverages=c(1,2,3,4,5,10), no_iterations=10, no_random_iterations
```

Arguments

data	MEDIPS SET
coverages	default is c(1, 2, 3, 4, 5, 10). The coverages define the depth levels for testing how often a CpG was covered by the given regions. Just specify any other vector of coverage depths you would like to test.
no_iterations	defines the number of subsets created from the full set of available regions (default=10).
no_random_iterations	approaches that randomly select data entries may be processed several times in order to obtain more stable results. By specifying the no_random_iterations parameter (default=1) it is possible to run the coverage analysis several times. The final results returned to the coverage results object are the averaged results of each random iteration step.
extend	extends the region lengths before the coverage analysis is performed.

Value

matrix	Contains the number of covered CpGs in each iteration (rows) and for different levels of coverages (columns)
maxPos	is the total number of sequence patterns (e.g. CpGs) within the refernce genome
pattern	is the defined sequence pattern
coveredPos	shows the number of covered sequence pattern (e.g. CpGs) using the total set of available regions for several depths of coverages (columns). The last row shows the percentage of covered sequence pattern relative to the total number of available sequence patterns within the reference genome.

Author(s)

Lukas Chavez

Examples

```
print("The function has been deprecated. Please see MEDIPS.seqCoverage.")
```

MEDIPS.CpGenrich *Calculates CpG enrichment of provided short reads compared to the reference genome.*

Description

As a quality check for the enrichment of CpG rich DNA fragments obtained by the immunoprecipitation step of a MeDIP experiment, this function 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, the function counts the number of Cs, the number of Gs, the number CpGs, and the total number of bases within the stated reference genome. Subsequently, the function calculates the relative frequency of CpGs and the observed/expected ratio of CpGs present in the reference genome. Additionally, the function 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.

Usage

```
MEDIPS.CpGenrich(file=NULL, BSgenome=NULL, extend=0, shift=0, uniq=TRUE, chr.select=NULL, paired=F)
```

Arguments

file	Path and file name of the input data
BSgenome	The reference genome name as defined by BSgenome
extend	defines the number of bases by which the region will be extended before the genome vector is calculated. Regions will be extended along the plus or the minus strand as defined by their provided strand information.
shift	As an alternative to the extend parameter, the shift parameter can be specified. 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.
uniq	MEDIPS will replace all reads which map to exactly the same start and end positions by only one representative, if uniq=TRUE
chr.select	only data at the specified chromosomes will be processed.
paired	option for paired end reads

Value

regions.CG	the numbe of CpGs within the regions
regions.C	the number of Cs within the regions
regions.G	the number of Gs within the regions
regions.relH	the relative frequency of CpGs within the regions
regions.GoGe	the observed/expected ratio of CpGs within the regions
genome.CG	the numbe of CpGs within the reference genome
genome.C	the number of Cs within the reference genome
genome.G	the number of Gs within the reference genome
genome.relH	the relative frequency of CpGs within the reference genome
genome.GoGe	the observed/expected ratio of CpGs within the reference genome
enrichment.score.relH	regions.relH/genome.relH
enrichment.score.GoGe	regions.GoGe/genome.GoGe

Author(s)

Joern Dietrich and Matthias Lienhard

Examples

```
library(MEDIPSData)
library("BSgenome.Hsapiens.UCSC.hg19")
bam.file.hESCs.Rep1.MeDIP = system.file("extdata", "hESCs.MeDIP.Rep1.chr22.bam", package="MEDIPSData")

#er=MEDIPS.CpGenrich(file=bam.file.hESCs.Rep1.MeDIP, BSgenome="BSgenome.Hsapiens.UCSC.hg19", chr.select="chr22"
```

MEDIPS.createROIset *Creates a MEDIPS ROI SET by reading a suitable input file*

Description

Reads the input file and calculates the short read coverage (counts) for the specified regions of interest(ROI). After reading of the input file, the MEDIPS ROI SET contains information about 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 a GRange object of the ROIs.

Usage

```
MEDIPS.createROIset(file=NULL, ROI=NULL, extend=0, shift=0, bn=1, BSgenome=NULL, uniq=TRUE, chr.select
```

Arguments

file	Path and file name of the input data
ROI	Data.frame with columns "chr", "start", "end" and "name" of regions of interest
extend	defines the number of bases by which the region will be extended before the genome vector is calculated. Regions will be extended along the plus or the minus strand as defined by their provided strand information.
shift	As an alternative to the extend parameter, the shift parameter can be specified. 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.
bn	Number of bins per ROI
BSgenome	The reference genome name as defined by BSgenome
uniq	MEDIPS will replace all reads which map to exactly the same start and end positions by only one representative, if uniq=TRUE
chr.select	only data at the specified chromosomes will be processed.
paired	option for paired end reads
sample_name	name of the sample to be stored with the MEDIPSROI SET.

Value

An object of class MEDIPSroiSet.

Author(s)

Lukas Chavez and Matthias Lienhard

Examples

```
library("BSgenome.Hsapiens.UCSC.hg19")
bam.file.hESCs.Rep1.MeDIP = system.file("extdata", "hESCs.MeDIP.Rep1.chr22.bam", package="MEDIPSData")
rois=data.frame(chr=c("chr22","chr22"), start=c(19136001, 19753401), stop=c(19136200, 19753500), ID=c("ID_1", "I
MSet=MEDIPS.createROIset(file=bam.file.hESCs.Rep1.MeDIP,ROI=rois, BSgenome="BSgenome.Hsapiens.UCSC.hg19", exten
```

MEDIPS.createSet *Creates a MEDIPS SET by reading a suitable input file*

Description

Reads the input file and calculates genome wide short read coverage (counts) at the specified window size. After reading of the input file, the MEDIPS SET contains information about the input file name, the dependent organism, the chromosomes included in the input file, the length of the included chromosomes (automatically loaded), and the number of regions.

Usage

```
MEDIPS.createSet(file=NULL, extend=0, shift=0, window_size=300, BSgenome=NULL, uniq=TRUE, chr.select=
```

Arguments

file	Path and file name of the input data
BSgenome	The reference genome name as defined by BSgenome
extend	defines the number of bases by which the region will be extended before the genome vector is calculated. Regions will be extended along the plus or the minus strand as defined by their provided strand information.
shift	As an alternative to the extend parameter, the shift parameter can be specified. 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.
uniq	MEDIPS will replace all reads which map to exactly the same start and end positions by only one representative, if uniq=TRUE
chr.select	only data at the specified chromosomes will be processed.
window_size	defines the genomic resolution by which short read coverage is calculated.
paired	option for paired end reads
sample_name	name of the sample to be stored with the MEDIPS SET.

Value

An object of class MEDIPSset.

Author(s)

Lukas Chavez and Mathias Lienhard

Examples

```
library("BSgenome.Hsapiens.UCSC.hg19")
bam.file.hESCs.Rep1.MeDIP = system.file("extdata", "hESCs.MeDIP.Rep1.chr22.bam", package="MEDIPSData")

MSet=MEDIPS.createSet(file=bam.file.hESCs.Rep1.MeDIP, BSgenome="BSgenome.Hsapiens.UCSC.hg19", chr.select="chr22
```

MEDIPS.exportWIG *Exports count, rpkm, or sequence pattern densities into a wiggle file.*

Description

The function allows for exporting the calculated methylation values (counts or rpkm) or sequence pattern densities from a MEDIPS or COUPLING SET into a wiggle (WIG) file. The wiggle file will contain values for all genomic windows of the genome/coupling vector and can be used for data visualization using appropriate genome browsers. Either a MEDIPS SET (parameter MSet) or a COUPLING SET (parameter CSet) has to be given.

Usage

```
MEDIPS.exportWIG(Set=NULL, file=NULL, format="rpkm", descr="")
```

Arguments

Set	has to be a MEDIPS SET or COUPLING SET object. In case of a MEDIPS SET, the parameter 'format' has to be either 'count' or 'rpkm'.
file	defines the name of the exported file
format	if set to "count", there must be a MEDIPS SET at 'Set', and the number of overlapping (extended) short reads per window will be exported. if set to "rpkm", there must be a MEDIPS SET at 'Set', and rpkm values will be exported (default). If set to "pdensity", there must be a COUPLING SET at 'Set', and the pattern densities (counts per window) will be exported.
descr	the exported wiggle file will include a track name and description that will be visualized by the utilized genome browser. Both, track name and description will be the same as defined here.

Value

the funtion exports the specified data from the MEDIPS or COUPLING SET into the stated file

Author(s)

Lukas Chavez

Examples

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

bam.file.hESCs.Rep1.MeDIP = system.file("extdata", "hESCs.MeDIP.Rep1.chr22.bam", package="MEDIPSData")
MSet=MEDIPS.createSet(file=bam.file.hESCs.Rep1.MeDIP, BSgenome="BSgenome.Hsapiens.UCSC.hg19", chr.select="chr22

MEDIPS.exportWIG(Set=MSet, file="hESCs.Rep1.wig", format="rpkm", descr="hESCs.Rep1")
```

`MEDIPS.genomeVector` *Calculates the genome wide short read coverage on a user specified resolution*

Description

This function has been deprecated. Please see MEDIPS.createSet instead.

Based on the regions included within a previously created MEDIPS SET (see MEDIPS.readAlignedSequiences), the function calculates the genome wide coverage on a user specified resolution. Each chromosome inside the MEDIPS SET will be divided into bins of size bin_size and the short read coverage will be calculated on this resolution. The bin representation of the genome is the 'genome vector'.

Usage

```
MEDIPS.genomeVector(data = NULL, extend = 400, bin_size = 50)
```

Arguments

`data`	MEDIPS SET
`extend`	Regions will be extended w.r.t. the extend parameter along the plus or the minus strand (as defined by their provided strand information). After extending the regions, their length will be 'extend' (i.e. the extend parameter is NOT added to the given read lengths but all regions will be of size 'extend' afterwards).
`bin_size`	defines the size of genome wide bins and therefore, the size of the genome vector. Read coverages will be calculated for bins separated by bin_size base pairs.

Value

The slots of the stated MEDIPS SET object associated to the genome vector will be occupied afterwards. These are the informations about the bin_size, the extend value, the chromosome and position of the bins, and the number regions within the MEDIPS SET that overlap with the genomic bins.

Author(s)

Lukas Chavez

Examples

```
print("This function has been deprecated. Please see MEDIPS.createSet instead.")
```

MEDIPS.getAnnotation *Funtion to fetch annotations from biomaRt.*

Description

The function receives predifined annotations from ensembl biomaRt for subsequent annotation of a result table.

Usage

```
MEDIPS.getAnnotation(host="www.biomart.org",dataset=c("hsapiens_gene_ensembl","mmusculus_gene_ensem
```

Arguments

host	BioMart database host you want to connect to. For current ensembl version, use "www.biomart.org". For other versions, set to the respective archive host, e.g. "may2012.archive.ensembl.org" for Ensembl 67 Please ensure that the ensembl version is compatible to the used genome version.
dataset	The dataset you want to use. To see the different datasets available within a biomaRt you can e.g. do: mart = useMart('ensembl'), followed by listDatasets(mart).
annotation	Type of annotation you want to retrieve. You can select "EXON" for exonic regions, "GENE" for gene regions including introns and "TSS" for regions at the transcirption start site.
tssSz	Defines the TSS region: start and end position relative to the strand and position of the transcript.
chr	Chromosome names for which the annotations should be filtered.

Value

The MEDIPS.getAnnotation function returns a list of annotation tables where each table consists of

id	the id of the annotation
chr	the chromosome of the annotation
start	the start position (5') of the annotation
end	the end position (3') of the annotation

Author(s)

Joern Dietrich and Matthias Lienhard

Examples

```
#homo sapiens, current ensembl version
#annotation_hs = MEDIPS.getAnnotation(dataset="hsapiens_gene_ensembl", annotation="TSS", chr=c("chr22"),tssSz=c(

#mus musculus, ensembl version 58 (mm9)
annotation_mm9 = MEDIPS.getAnnotation(host="may2010.archive.ensembl.org",dataset="mmusculus_gene_ensembl", anno
```

`MEDIPS.mergeFrames` *Merges genomic coordinates of neighboring windows into one supersized window*

Description

After having filtered the result table returned by the MEDIPS.meth function using the MEDIPS.selectSig function, there might be neighboring significant frames. For these cases it is worthwhile to merge neighbouring regions into one supersized frame.

Usage

```
MEDIPS.mergeFrames(frames=NULL, distance=1)
```

Arguments

`frames`	is a filtered result table received by the MEDIPS.selectSig function.
`distance`	allows an according number of bases as a gap between neighboring significant windows to be merged. The default value is 1 in order to merge adjacent windows.

Value

The remaining distinct frames are represented only by their genomic coordinates within the returned results table

`chromosome`	the chromosome of the merged frame
`start`	the start position of the merged frame
`stop`	the stop position of the merged frame
`ID`	a numbered ID of the merged frame

The result table does not contain any merged significant values.

Author(s)

Lukas Chavez

Examples

```
regions=as.data.frame(list(chr=c("chr22", "chr22"), start=c(1001, 1501), stop=c(1500,1750)))
regions.merged=MEDIPS.mergeFrames(regions)

regions.merged
```

MEDIPS.mergeSets *Creates one merged MEDIPS SET out of two.*

Description

A MEDIPS SET contains a genome vector which is the count coverage at genome wide windows. Moreover, the MEDIPS SET stores the total number of reads given for calculating the genome vector. Two MEDIPS SETs can be merged whenever they have been construcetd based on the same reference genome, the same set of chromosomes and for the same window size. The returned MEDIPS SET will contain a genome vector where at each window the counts of both given MEDIPS SETs are added. In addtion, the total number of reads will be the sum of both given MEDIPS SETs. Please note, several other attributes like the extend or shift value can be different in both of the given MEDIPS SETs and will be empty in the merged MEDIPS SET. The merged MEDIPS SET will not contain any path to a concrete input file anymore and therefore, cannot be used for the MEDIPS.addCNV function anymore.

Usage

```
MEDIPS.mergeSets(MSet1=NULL, MSet2=NULL, name="Merged Set")
```

Arguments

MSet1	A MEDIPS SET object as created by the MEDIPS.createSet function
MSet2	A MEDIPS SET object as created by the MEDIPS.createSet function
name	The new sample name of the merged MEDIPS SET

Author(s)

Lukas Chavez

Examples

```
library(MEDIPSData)
data(hESCs_Input)
data(DE_Input)

merged_Set = MEDIPS.mergeSets(hESCs_Input, DE_Input, name="Merged_input")

merged_Set
```

`MEDIPS.meth` *Funtion summarizes coverage profiles for given MEDIPS SETs and allows to calculate differental coverage and copy number vartiation, if applicable.*

Description

The function summarizes coverage profiles (counts, rpkm) for given MEDIPS SETs at the slots MSet1, MSet2, ISet1, and ISet1. In case the parameter MeDIP is set to TRUE and a COUPLING SET was provided at the slot CS, the function will calculate normalized methylation profiles (rms, prob) for the MEDIPS SETs at the slots MSet1 and MSet2. In case two groups of MEDIPS SETs have been provided at MSet1 and MSet2, the function will calculate differential coverage. In case two groups of MEDIPS SETs have been provided at ISet1 and ISet2 and the parameter CNV was set to TRUE, the function will calculate copy number variation. Because the function allows for processing a variable number of provided MEDIPS SETs, the returned matrix is of variable length.

Usage

```
MEDIPS.meth(MSet1 = NULL, MSet2 = NULL, CSet = NULL, ISet1 = NULL, ISet2 = NULL, chr = NULL, p.adj="bonfe
```

Arguments

`MSet1`	has to be one or a concatenated list of MEDIPS SET objects (the control replicates)
`MSet2`	has to be one or a concatenated list of MEDIPS SET objects (the treatment data) or empty
`CSet`	has to be a COUPLING SET object (must fit the given MEDIPS SET objects with respect to reference genome and represented chromosomes)
`ISet1`	has to be one or a concatenated list of Input derived MEDIPS SET objects (general Input data or Inputs from the control replicates) or empty
`ISet2`	has to be one or a concatenated list of Input derived MEDIPS SET objects (Inputs from the treatment replicates) or empty
`chr`	specify one or several chromosomes (e.g. c("chr1", "chr2")), if only a subset of available chromosomes have to be processed.
`p.adj`	in order to correct p.values derived from the differential coverage analysis for multiple testing, MEDIPS uses Rs' p.adjust function. Therefore, the following methods are available: holm, hochberg, hommel, bonferroni (default) , BH, BY, fdr, none.
`diff.method`	method for calculating differential coverage. Available methods: ttest (default) and edgeR.
`prob.method`	Provided that the parameter MeDIP is set to TRUE, MEDIPS will calculate CpG density dependent probability values in order to estimate the methylation status of genome wide windows. For this, MEDIPS calculates a series of CpG coupling factor dependent probability distributions. The methylation status of

each window will be estimated by its according distribution. There are two probability distributions available: poisson (default) and negBinomial. Please note that we consider this method to be poorly conceived and under further development.

`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 (default: CNV=TRUE). Please note, there is the function MEDIPS.addCNV which allows to run the CNV analysis on two groups of INPUT SETs using another (typically increased) window size.

`MeDIP` This parameter determines, whether for the MEDIPS SETs given at the slots MSet1 and MSet2, CpG density dependent normalization values (rms and prob) will be calculated (default: MeDIP=TRUE).

`type` In case diff.method has been set to ttest, this parameter specifies, if differential coverage is calculated based on the rpkm (default) or rms values. This parameter is ignored in case the edgeR method is selected as the underlying model requires count data.

`minRowSum` threshold for the sum of counts in a window for the staistical test (default=1).

Value

`Chr` the chromosome of the ROI

`Start` the start position of the ROI

`Stop` the stop position of the ROI

`CF` the number of CpGs in the window

`*counts` a variable number of columns (according to the number of provided MEDIPS SETs) containing for each set the number of (extended/shifted) reads that overlap with the window.

`*rpkm` a variable number of columns (according to the number of provided MEDIPS SETs) containing for each set the rpkm value of the window.

`*rms` optional (if MeDIP=TRUE): a variable number of columns (according to the number of provided MEDIPS SETs) containing for each set the rms value of the window.

`*prob` optional (if MeDIP=TRUE): a variable number of columns (according to the number of provided MEDIPS SETs) containing for each set the probability of methylation [0:1] of the window.

`*counts` optional (if INPUT SETs given): a variable number of columns (according to the number of provided INPUT SETs) containing for each set the number of (extended/shifted) reads that overlap with the window.

`*rpkm` optional (if INPUT SETs given): a variable number of columns (according to the number of provided INPUT SETs) containing for each set the rpkm value of the window.

`MSets1.counts.mean`
optional (if more than one MEDIPS SET given): the mean count over all MEDIPS SETs at MSet1.

`MSets1.rpkm.mean`
: optional (if more than one MEDIPS SET given): the mean rpkm value over all MEDIPS SETs at MSet1.

`MSets1.rms.mean`
: optional (if more than one MEDIPS SET given): the mean rms value over all MEDIPS SETs at MSet1.

`MSets1.prob.mean`
: optional (if more than one MEDIPS SET given): the mean probability value over all MEDIPS SETs at MSet1.

`MSets2.counts.mean`
: optional (if more than one MEDIPS SET given): the mean count over all MEDIPS SETs at MSet2.

`MSets2.rpkm.mean`
: optional (if more than one MEDIPS SET given): the mean rpkm value over all MEDIPS SETs at MSet2.

`MSets2.rms.mean`
: optional (if more than one MEDIPS SET given): the mean rms value over all MEDIPS SETs at MSet2.

`MSets2.prob.mean`
: optional (if more than one MEDIPS SET given): the mean probability value over all MEDIPS SETs at MSet2.

`ISets1.counts.mean`
: optional (if more than one INPUT SET given): the mean count over all INPUT SETs at ISet1.

`ISets1.rpkm.mean`
: optional (if more than one INPUT SET given): the mean rpkm value over all INPUT SETs at ISet1.

`ISets2.counts.mean`
: optional (if more than one INPUT SET given): the mean count over all INPUT SETs at ISet2.

`ISets2.rpkm.mean`
: optional (if more than one INPUT SET given): the mean rpkm value over all INPUT SETs at ISet2.

`edgeR.logFC` optional (if diff.method=edgeR): log fold change between MSet1 and MSet2 as returned by edgeR.

`edgeR.logCPM` optional (if diff.method=edgeR): logCPM between MSet1 and MSet2 as returned by edgeR.

`edgeR.p.value` optional (if diff.method=edgeR): p.value as returned by edgeR.

`edgeR.adj.p.value`
: optional (if diff.method=edgeR): adjusted p.value as calculated by the p.adjust function using edgeR's p.values as input.

`score.log2.ratio`
: optional (if diff.method=ttest): log2 fold change between the means of the groups MSet1 and MSet2.

`score.p.value` optional (if diff.method=ttest): p.value as returned by the t.test function.

`score.adj.p.value`
: optional (if diff.method=ttest): adjusted p.value as calculated by the p.adjust function using the ttest p.values as input.

score	optional (if diff.method=ttest): score = (-log10(score.p.value)*10)*log(score.log2.ratio)
CNV.log2.ratio	optional (if two INPUT SETs given and CNV=TRUE): the log2 ratio for segments as calculated by the DNAcopy package.

Author(s)

Lukas Chavez, Matthias Lienhard, Joern Dietrich

Examples

```
library(MEDIPSData)
data(hESCs_MeDIP)
data(DE_MeDIP)
data(hESCs_Input)
data(DE_Input)
data(CS)

resultTable = MEDIPS.meth(MSet1 = hESCs_MeDIP, MSet2 = DE_MeDIP, CSet = CS, ISet1 = hESCs_Input, ISet2 = DE_Input, c
```

MEDIPS.methylProfiling

Funtion calculates mean methylation values (rpm, rms, ams), ratios, variances, and p-values comparing two MEDIPS SETs for user supplied regions of interests (ROIs) or for genome wide frames.

Description

This function has been deprecated. Please see MEDIPS.meth instead.

In order to compare two different conditions, first you have to create and process two sets of MEDIPS SETs. For the identification of DMRs, MEDIPS provides two alternative approaches. First, you can specify pre-defined regions of interest (ROIs). Second, MEDIPS offers the possibility to calculate differential methylation for genome wide frames. The function calculates summarized methylation values for the defined ROIs. Here, these are the mean values for the provided MEDIPS SETs as well as the ratio of means. Moreover, for each ROI, MEDIPS calculates p-values by comparing the set of rpm values (or rms values, respectively) within the ROI of the one MEDIPS SET against the set of rpm values (or rms values, respectively) within the ROI of the second MEDIPS SET using R's wilcox.test and t.test functions. Additionally, it is recommended (but not necessary) to provide background data from an INPUT experiment (that is sequencing of none-enriched DNA fragments). By providing an INPUT data set, MEDIPS additionally returns mean INPUT rpm values for the specified ROIs. Becuase the function allows for processing a variable number of provided MEDIPS SETs, the returned matrix is of variable length.

Usage

```
MEDIPS.methylProfiling(data1 = NULL, data2 = NULL, input = NULL, ROI_file = NULL, frame_size = NULL, mat
```

Arguments

data1	MEDIPS SET control
data2	MEDIPS SET treatment
input	INPUT SET
ROI_file	instead of processing genome wide frames using the parameters frame_size and step, here you can provide a file containing predefined ROIs.
frame_size	Besides summarizing methylation values for pre-defined ROIs, MEDIPS allows for calculating mean methylation values along the full chromosomes. For this, you have to specify a desired frame size here.
math	default=mean; Here, you can specify other functions available in R for sumarizing values like median or sum.
step	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.
select	can be either 1 or 2. If set to 1, the variance, ratio, and p-values will be calculated based on the rpm values; if set to 2, the rms values will be considered instead.
chr	only the specified chromosome will be evaluated (e.g. chr1)
transf	transforms the resulting data range into the interval [0:1000]

Value

Chr	the chromosome of the ROI
Start	the start position of the ROI
Stop	the stop position of the ROI
Items	the number of genomic bins included in the ROI
CF	the mean coupling factor of the ROI
RPM_MSet1.*	the mean reads per million value for the MEDIPS SET at position * of MSet1
RMS_MSet1.*	the mean relative methylation score for the MEDIPS SET at position * of MSet1
AMS_MSet1.*	the mean absolute mathylation score for the MEDIPS SET at position * of MSet1
RPM_MSet2.*	the mean reads per million value for the MEDIPS SET at position * of MSet2 (if provided)
RMS_MSet2.*	the mean relative methylation score for the MEDIPS SET at position * of MSet2 (if provided)
AMS_MSet2.*	the mean absolute mathylation score for the MEDIPS SET at position * of MSet2 (if provided)
RPM_ISet1.*	the mean reads per million value of the Input MEDIPS SET at position * of ISet1 (if provided)
RPM_ISet2.*	the mean reads per million value of the Input MEDIPS SET at position * of ISet2 (if provided)

RPM_MSets1	the mean reads per million value over all MEDIPS SETs at MSet1
RMS_MSets1	the mean relative methylation score over all MEDIPS SETs at MSet1
AMS_MSets1	the mean absolute methylation score over all MEDIPS SETs at MSet1
RPM_MSets2	the mean reads per million value over all MEDIPS SETs at MSet2
RMS_MSets2	the mean relative methylation score over all MEDIPS SETs at MSet2
AMS_MSets2	the mean absolute methylation score over all MEDIPS SETs at MSet2
RPM_ISets1	the mean reads per million value over all MEDIPS SETs at ISet1
RPM_ISets2	the mean reads per million value over all MEDIPS SETs at ISet2
V_MSet1	the variance of the rpm or rms values (please see the parameter select) over all MEDIPS SETs at MSet1
CV_MSet1	the coefficient of variance of the rpm or rms values (please see the parameter select) over all MEDIPS SETs at MSet1
V_MSet2	the variance of the rpm or rms values (please see the parameter select) of all MEDIPS SETs at MSet2 (if provided)
CV_MSet2	the coefficient of variance of the rpm or rms values (please see the parameter select) over all MEDIPS SETs at MSet2 (if provided)
Ratio	RPM_MSets1/RPM_MSets2 or RMS_MSets1/RMS_MSets2, respectively (please see the parameter select)
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 SETs at MSet1 and of the MEDIPS SETs at MSet2
T.test	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 SETs at MSet1 and of the MEDIPS SETs at MSet2
Wilcox.adj	multiple testing corrected p.value from slot Wilcox; calculated by R's p.adjust function
T.test.adj	multiple testing corrected p.value from slot T.Test; calculated by R's p.adjust function

Author(s)

Joern Dietrich

Examples

```
print("This function has been deprecated. Please see MEDIPS.meth instead.")
```

`MEDIPS.normalize` *Function that normalizes raw signals by local sequence pattern (e.g. CpG) densities.*

Description

This function has been deprecated. Please see MEDIPS.rms instead.

The normalization function accesses the pre-calculated slope and intercept values derived from the MEDIPS.calibrationCurve function in order to weight the raw signals. The relative methlyation score (rms) for the genomic bins is then defined by rms = x\((y-intercept)/slope), where x is the raw signal and y is the coupling factor of a genomic bin. Based on the total number of regions within the MEDIPS SET, the rms values will be transformed into a reads per million format.

Usage

```
MEDIPS.normalize(data=NULL)
```

Arguments

`data`	has to be a MEDIPS SET object

Value

RMS signals.

Author(s)

Lukas Chavez

Examples

```
print("This function has been deprecated. Please see MEDIPS.rms instead.")
```

`MEDIPS.plotCalibrationPlot` *Creates the calibration plot.*

Description

Visualizes the dependency between MeDIP-seq signals and CpG densities together with the calcibration curve and the results of the linear modelling.

Usage

```
MEDIPS.plotCalibrationPlot(MSet=NULL, ISet=NULL, CSet=NULL, plot_chr="chr1", rpkm=T, main="Calibratio
```

Arguments

MSet	a MEDIPS SET object
ISet	an INPUT SET (i.e. a MEDIPS SET created from Input sequening data)
CSet	an according COUPLING SET object
plot_chr	default="chr1". It is recommended to call a graphics device (e.g. png("calibrationPlot.png")) before calling the plot command, because R might not be able to plot the full amount of data in reasonable time.
rpkm	can be either TRUE or FALSE. If set to TRUE, the methylation signals will be transformed into rpkm before plotted. The coupling values remain untouched. It is necessary to set rpkm=T, if both, a MSet and an ISet are given and plotted at the same time.
main	The title of the calibration plot.
xrange	The signal range of the calibration curve typically falls into a low signal range. By setting the xrange parameter to TRUE, the calibration plot will visualize the low signal range only.

Value

The calibration plot will be visualized.

Author(s)

Lukas Chavez, Matthias Lienhard

Examples

```
library(MEDIPSData)
data(hESCs_MeDIP)
data(CS)

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

MEDIPS.plotCoverage *Function plots the results of the MEDIPS.coverageAnalysis function.*

Description

This function has been deprecated. Please see MEDIPS.plotSeqCoverage instead.

The results of the coverage analysis will be visualized by the function.

Usage

```
MEDIPS.plotCoverage(coverageObj = NULL)
```

Arguments

coverageObj	The coverage results object returned by the MEDIPS.coverageAnalysis function.

Value

The coverage plot will be visualized.

Author(s)

Lukas Chavez

Examples

```
print("This function has been deprecated. Please see MEDIPS.plotSeqCoverage instead.")
```

MEDIPS.plotSaturation *Function plots the results of the MEDIPS.saturationAnalysis function.*

Description

The results of the saturation analysis will be visualized by the function.

Usage

```
MEDIPS.plotSaturation(saturationObj = NULL, main="Saturation analysis")
```

Arguments

saturationObj	The saturation results object returned by the MEDIPS.saturationAnalysis function
main	The title of the coverage plot.

Value

The coverage plot will be visualized.

Author(s)

Lukas Chavez

Examples

```
library(MEDIPSData)
library(BSgenome.Hsapiens.UCSC.hg19)
bam.file.hESCs.Rep1.MeDIP = system.file("extdata", "hESCs.MeDIP.Rep1.chr22.bam", package="MEDIPSData")

sr=MEDIPS.saturation(file=bam.file.hESCs.Rep1.MeDIP, BSgenome="BSgenome.Hsapiens.UCSC.hg19", uniq=TRUE, extend=
MEDIPS.plotSaturation(saturationObj = sr, main="Saturation analysis")
```

MEDIPS.plotSeqCoverage

Function plots the results of the MEDIPS.seqCoverage function.

Description

The results of the sequence pattern coverage analysis will be visualized in two possible ways.

Usage

```
MEDIPS.plotSeqCoverage(seqCoverageObj=NULL, main=NULL, type="pie", cov.level = c(0,1,2,3,4,5), t="Inf
```

Arguments

seqCoverageObj	The coverage results object returned by the MEDIPS.seqCoverage function.
main	The title of the coverage plot.
type	there are two types of visualization. The pie chart (default) illustrates the fraction of CpGs covered by the given reads at different coverage level (see also the parameter cov.level). As an alternative, a histogram over all coverage level can be ploted ("hist").
cov.level	The pie chart illustrates the fraction of CpGs covered by the given reads according to their coverage level. The visualized coverage levels can be adjusted by the cov.level parameter.
t	specifies the maximal coverage depth to be plotted, if type="hist"

Value

The sequence pattern coverage plot will be visualized.

Author(s)

Lukas Chavez

Examples

```
library(MEDIPSData)
library(BSgenome.Hsapiens.UCSC.hg19)
bam.file.hESCs.Rep1.MeDIP = system.file("extdata", "hESCs.MeDIP.Rep1.chr22.bam", package="MEDIPSData")

cr=MEDIPS.seqCoverage(file=bam.file.hESCs.Rep1.MeDIP, pattern="CG", BSgenome="BSgenome.Hsapiens.UCSC.hg19", chr

MEDIPS.plotSeqCoverage(seqCoverageObj=cr, main="Sequence pattern coverage", type="pie", cov.level = c(0,1,2,3,4,5
```

MEDIPS.readAlignedSequences

Creates a MEDIPS SET by reading a suitable input file

Description

This function has been deprecated. Please see MEDIPS.createSet instead.

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

Usage

```
MEDIPS.readAlignedSequences(file = NULL, BSgenome = NULL, numrows = -1)
```

Arguments

file	Path and file name of the input data
BSgenome	The reference genome name as defined by BSgenome
numrows	The number of short reads (number of rows) within the input file

Value

An object of class MEDIPSset is returned where the region dependent informations are stored in the according slots. These are informations about the input file, the reference genome, the total number of provided regions, the chromosomes which are covered by the regions, the total chromosome lengths, and the start and stop positions and strand informations of the regions.

Author(s)

Lukas Chavez

Examples

```
print("This function has been deprecated. Please see MEDIPS.createSet instead.")
```

`MEDIPS.saturation` *Function calculates the saturation/reproducibility of the provided IP-Seq data.*

Description

The saturation analysis addresses the question, whether the number of short reads is sufficient to generate a saturated and reproducible coverage profile of the reference genome. The main idea is that an insufficent 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 coverage profile will be reproducible by another independent set of a similar number of short reads.

Usage

```
MEDIPS.saturation(file=NULL, BSgenome=NULL, nit=10, nrit=1, empty_bins=TRUE, rank=FALSE, extend=0, shi
```

Arguments

`file`	Path and file name of the IP data
`BSgenome`	The reference genome name as defined by BSgenome
`nit`	defines the number of subsets created from the full sets of available regions (default=10)
`nrit`	methods which 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 each random iteration step.
`empty_bins`	can be either TRUE or FALSE (default TRUE). This parameter effects the way of calculating correlations between the resulting genome vectors. A genome vector consists of concatenated vectors for each included chromosome. The size of the vectors is defined by the bin_size parameter. If there occur genomic bins which contain no overlapping regions, neither from the subsets of A nor from the subsets of B, these bins will be neglected when the 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 windows instead of considering the counts (Spearman correlation). Setting this parameter to TRUE is a more robust approach that reduces the effect of possible occuring outliers (these are windows with a very high number of overlapping regions) to the correlation.

extend	defines the number of bases by which the region will be extended before the genome vector is calculated. Regions will be extended along the plus or the minus strand as defined by their provided strand information. Please note, the extend and shift parameter are mutual exclusive.
shift	defines the number of bases by which the region will be shifted before the genome vector is calculated. Regions will be shifted along the plus or the minus strand as defined by their provided strand information. Please note, the extend and shift parameter are mutual exclusive.
window_size	defines the size of genome wide windows and therefore, the size of the genome vector.
uniq	if uniq=TRUE (default), all reads with exactly the same start and end positions will be replaced by one representative before the saturation analysis is performed.
chr.select	specify a subset of chromosomes for which the saturation analysis is performed.
paired	option for paired end reads

Value

distinctSets	Contains the results of each iteration step (row-wise) of the saturation analysis. The first column is the number of considered regions in each set, the second column is the resulting pearson correlation coefficient when comparing the two independent genome vectors.
estimation	Contains the results of each iteration step (row-wise) of the estimated saturation analysis. The first column is the number of considered regions in each set, the second column is the resulting pearson correlation coefficient when comparing the two independent genome vectors.
distinctSets	the total number of available regions
maxEstCor	contains the best pearson correlation (second column) obtained by considering the artifically doubled set of reads (first column)
distinctSets	contains the best pearson correlation (second column) obtained by considering the total set of reads (first column)

Author(s)

Lukas Chavez

Examples

```
library(MEDIPSData)
library(BSgenome.Hsapiens.UCSC.hg19)
bam.file.hESCs.Rep1.MeDIP = system.file("extdata", "hESCs.MeDIP.Rep1.chr22.bam", package="MEDIPSData")

sr=MEDIPS.saturation(file=bam.file.hESCs.Rep1.MeDIP, BSgenome="BSgenome.Hsapiens.UCSC.hg19", uniq=TRUE, extend=
```

MEDIPS.saturationAnalysis

Function calculates the saturation/reproducibility of the provided MeDIP-Seq data.

Description

This function has been deprecated. Please see MEDIPS.saturation instead.

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 insufficent 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.

Usage

```
MEDIPS.saturationAnalysis(data=NULL, no_iterations=10, no_random_iterations=1, empty_bins=TRUE, rank=
```

Arguments

data	MEDIPS SET
no_iterations	defines the number of subsets created from the full sets of available regions (default=10)
no_random_iterations	approaches that randomly select data entries may be processed several times in order to obtain more stable results. By specifying the no_random_iterations parameter (default=1) it is possible to run the saturation analysis several times. The final results returned to the saturation results object are the averaged results of each random iteration step.
empty_bins	can be either TRUE or FALSE (default TRUE). This parameter effects the way of calculating correlations between the resulting genome vectors. A genome vector consists of concatenated vectors for each included chromosome. The size of the vectors is defined by the bin_size parameter. If there occur genomic bins which contain no overlapping regions, neither from the subsets of A nor from the subsets of B, these bins will be neglected when the 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.
extend	defines the number of bases by which the region will be extended before the genome vector is calculated. Regions will be extended along the plus or the minus strand as defined by their provided strand information.

`bin_size` defines the size of genome wide bins and therefore, the size of the genome vector. Read coverages will be calculated for bins separated by bin_size base pairs.

Value

`distinctSets` Contains the results of each iteration step (row-wise) of the saturation analysis. The first column is the number of considered regions in each set, the second column is the resulting pearson correlation coefficient when comparing the two independent genome vectors.

`estimation` Contains the results of each iteration step (row-wise) of the estimated saturation analysis. The first column is the number of considered regions in each set, the second column is the resulting pearson correlation coefficient when comparing the two independent genome vectors.

`distinctSets` the total number of available regions

`maxEstCor` contains the best pearson correlation (second column) obtained by considering the artifically doubled set of reads (first column)

`distinctSets` contains the best pearson correlation (second column) obtained by considering the total set of reads (first column)

Author(s)

Lukas Chavez

Examples

```
print("This function has been deprecated. Please see MEDIPS.saturation instead")
```

MEDIPS.selectROIs *Selects row-wise subsets of a result table as returned by the MEDIPS.meth function.*

Description

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).

Usage

```
MEDIPS.selectROIs(results=NULL, rois=NULL, columns=NULL, summarize=NULL)
```

Arguments

results	a result table as returned by the function MEDIPS.meth
rois	A matrix containing genomic coordinates of any regions of interest.
columns	Only selected columns will be returned as determined by the columns parameter. It is possible to specify one or more concrete column names, please see an example below.
summarize	By setting summarize=NULL (default) all windows included in the genomic ranges of the given ROIs will be returned. As an alternative, it is possible to calculate mean values over the individual windows included in each ROI (summarize = "avg"), or to select only the most significant window within each given ROI (summarize="minP").

Author(s)

Lukas Chavez, Matthias Lienhard

Examples

```
library(MEDIPSData)
data(resultTable)

rois=data.frame(chr=c("chr22","chr22"), start=c(19136001, 19753401), stop=c(19136200, 19753500), ID=c("ID_1", "I
columns=names(resultTable)[grep("counts|rpkm|logFC",names(resultTable))]
s = MEDIPS.selectROIs(results=resultTable, rois=rois, columns=columns, summarize=NULL)
```

MEDIPS.selectSig — *Selects windows which show significant differential coverage between two MEDIPS SETs from a resultTable (as returned by the function MEDIPS.meth).*

Description

Based on the results table returned by the MEDIPS.meth function, the function selects windows which show significant differential coverage between the two groups of MEDIPS SETs. Selection of significant windows follows according to the specification of the available parameters.

Usage

```
MEDIPS.selectSig(results=NULL, p.value=0.01, adj=T, ratio=NULL, bg.counts=NULL, CNV=F)
```

Arguments

results	specifies the result table derived from the MEDIPS.meth function.
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. For this, the mean of counts per group is considered. The parameter bg.counts can either be a concrete integer or an appropriate column name of the result table. By specifying a column name, 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 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 subtract the CNV ratio from the IP ratio. Subsequently, 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) will be considered as windows with sufficient differential IP coverage.

Author(s)

Lukas Chavez, Matthias Lienhard

Examples

```
library(MEDIPSData)
data(resultTable)

sig = MEDIPS.selectSig(results=resultTable, p.value=0.05, adj=TRUE, ratio=NULL, bg.counts=NULL, CNV=FALSE)
```

MEDIPS.selectSignificants

Selects candidate ROIs that show significant differential methylation between two MEDIPS SETs.

Description

This function has been deprecated. Please see MEDIPS.selectSig instead.

Based on the results matrix returned from the MEDIPS.diffMethyl function, the function selects 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. Filtering for significant frames proceeds in the following order: ROIs that do not contain any data either in the CONTROL.SET nor in the TREAT.SET are neglected first; ROIs associated to p-values > p.value are neglected; ROIs with a CONTROL/TREATMENT ratio < up (or > down, respectively) are neglected; From the INPUT mean rpm distribution, a mean rpm threshold was defined by the quant parameter and all ROIs that have a mean rpm value within the CONTROL.SET (or TREAT.SET, respectively) smaller than the estimated background rpm threshold are discarded; The last filter is again based on the INPUT data. While the latter filter estimates a minimum rpm signal for the CONTROL.SET (or TREAT.SET, respectively) from the total background distribution, we now define that the rpm value from the CONTROL SET (or TREAT.SET, respectively) of a ROI exceeds the local background data of the INPUT.SET by the parameter up. This is, because MeDIP-Seq background data varies along the chromosomes due to varying DNA availability.

Usage

```
MEDIPS.selectSignificants(frames = NULL, input = T, control = T, up = 1.333333, down = 0.75, p.value = 0.
```

Arguments

frames	specifies the results table derived from the MEDIPS.diffMethyl
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 threshold during the subsequent analysis. How to estimate such a threshold without background data is not yet solved by MEDIPS.
control	can be either TRUE or FALSE; MEDIPS allows for selecting frames 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=T, MEDIPS selects genomic regions, where the CONTROL SET is higher methylated. By setting control=F, MEDIPS selects genomic regions, where the TREAT SET is higher methylated.
up	default=1.333333; defines the lower threshold for the ratio CONTROL/TREAT as well as for 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 p-values derived from the wilcox.test or t.test function has to be <= p.value.
quant	default=0.9; from the distribution of all summarized INPUT rpm values, MEDIPS calculates the rpm value that represents the quant quantile of the whole INPUT distribution.

Value

chr	the chromosome of the ROI
start	the start position of the ROI
stop	the stop position of the ROI
length	the number of genomic bins included in the ROI
coupling	the mean coupling factor of the ROI
input	the mean reads per million value of the INPUT MEDIPS SET at input (if provided)
rpm_A	the mean reads per million value for the MEDIPS SET at data1
rpm_B	the mean reads per million value for the MEDIPS SET at data2
rms_A	the mean relative mathylation score for the MEDIPS SET at data1
rms_B	the mean relative methylation score for the MEDIPS SET at data2
ams_A	the mean absolute mathylation score for the MEDIPS SET at data1. The ams scores are derived by dividing the mean rms value of the ROI by the mean coupling factor of the ROI before the log2 and interval transformations are performed.
ams_B	the mean absolute mathylation score for the MEDIPS SET at data2. The ams scores are derived by dividing the mean rms value of the ROI by the mean coupling factor of the ROI before the log2 and interval transformations are performed.
var_A	the variance of the rpm or rms values (please see the parameter select) of the MEDIPS SET at data1
var_B	the variance of the rpm or rms values (please see the parameter select) of the MEDIPS SET at data2
var_co_A	the variance coefficient of the rpm or rms values (please see the parameter select) of the MEDIPS SET at data1
var_co_B	the variance coefficient of the rpm or rms values (please see the parameter select) of the MEDIPS SET at data2
ratio	rpm_A/rpm_B or rms_A/rms_B, respectively (please see the parameter select)
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 at data1 and of the MEDIPS SET at data2
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 at data1 and of the MEDIPS SET at data2

Author(s)

Lukas Chavez

Examples

```
print("This function has been deprecated. Please see MEDIPS.selectSig instead.")
```

MEDIPS.seqCoverage *The function identifies the number of CpGs (or any other predefined sequence pattern) covered by the given short reads.*

Description

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 and to test the depth of coverage.

Usage

```
MEDIPS.seqCoverage(file = NULL, BSgenome = NULL, pattern = "CG", extend = 0, shift = 0, uniq = TRUE, chr.
```

Arguments

file	Path and file name of the input data
BSgenome	The reference genome name as defined by BSgenome
pattern	defines the sequence pattern, e.g. CG for CpGs.
extend	defines the number of bases by which the region will be extended before the genome vector is calculated. Regions will be extended along the plus or the minus strand as defined by their provided strand information. Please note, the extend and shift parameter are mutual exclusive.
shift	defines the number of bases by which the region will be shifted before the genome vector is calculated. Regions will be shifted along the plus or the minus strand as defined by their provided strand information. Please note, the extend and shift parameter are mutual exclusive.
uniq	if uniq=TRUE (default), all reads with exactly the same start and end positions will be replaced by one representative before the saturation analysis is performed.
chr.select	specify a subset of chromosomes for which the saturation analysis is performed.
paired	option for paired end reads

Author(s)

Lukas Chavez

Examples

```
library(MEDIPSData)
library(BSgenome.Hsapiens.UCSC.hg19)
bam.file.hESCs.Rep1.MeDIP = system.file("extdata", "hESCs.MeDIP.Rep1.chr22.bam", package="MEDIPSData")

cr = MEDIPS.seqCoverage(file=bam.file.hESCs.Rep1.MeDIP, BSgenome="BSgenome.Hsapiens.UCSC.hg19", pattern="CG", ex
```

MEDIPS.setAnnotation *Funtion to annotate a matrix of genomic coordinates (i.e. a result table) by a given annotation object.*

Description

The function appends any annotation IDs included in the given annotation object to the given regions object. An annotation object can be retrived by the MEDIPS.getAnnotation function and the regions object is typically a (filtered) result table as returned by the MEDIPS.meth function. An annotation ID is appended to a genomic region if their genomic coordinates overlap by at least one base. There will be as many columns added to the regions object as overlapping annotations exist in the annotation object.

Usage

```
MEDIPS.setAnnotation(regions, annotation, cnv=F)
```

Arguments

regions	a matrix that contains row-wise genomic regions, e.g. as a result of the MEDIPS.meth function.
annotation	the annotation data object contains the genomic coordinates of annotations. An annotation object can be e.g. retrived by the MEDIPS.getAnnotation function.
cnv	the MEDIPS.setAnnotation function is also internally used by the MEDIPS.addCNV function which automatically sets this parameter to TRUE. Otherwise cnv should be set to FALSE.

Value

The provided result object with added columns containing overlapping annotations.

Author(s)

Joern Dietrich, Matthias Lienhard

Examples

```
library(MEDIPSData)
data(resultTable)

sig = MEDIPS.selectSig(results=resultTable, p.value=0.05, adj=TRUE, ratio=NULL, bg.counts=NULL, CNV=FALSE)
sig = MEDIPS.mergeFrames(frames=sig, distance=1)
ens_gene = MEDIPS.getAnnotation( annotation="GENE",chr="chr22")
sig = MEDIPS.setAnnotation(regions=sig, annotation=ens_gene)
```

MEDIPSroiSet-class *MEDIPSroiSet class and internal functions*

Description

MEDIPSroiSet class is used in the MEDIPS library in order to store and extract objects and information of the specified regions of interest (ROI) from the input file as well as parameter settings specified during the workflow.

Objects from the Class

Objects of the classes contain information about the provided short reads, MeDIP raw/count signals, and further parameter settings. A MEDIPS ROI SET object is created by the MEDIPS.createROIset() function. According slots will be filled during the workflow.

Slots

`genome_name`**:** Object of class `"character"` : the refernce genome

`chr_names`**:** Object of class `"character"` : the names of the chromosomes included within the MEDIPS ROI SET

`chr_lengths`**:** Object of class `"numeric"` : the lengths of the chromosomes included within the MEDIPS ROI SET

`sample_name`**:** Object of class `"character"` : the name of the input file

`path_name`**:** Object of class `"character"` : the path to the input file

`number_regions`**:** Object of class `"numeric"` : the total number of included regions

`genome_count`**:** Object of class `"numeric"` : the raw MeDIP-seq signals at the bins

`extend`**:** Object of class `"numeric"` : the length of the reads after extension

`shifted`**:** Object of class `"numeric"` : the number of bases by which the reads are shifted along the sequencing direction

`uniq`**:** Object of class `"logical"` : determines if reads mapping to exactly the same genomic position should be replaced by only on representative

`ROI`**:** Object of class `"GRanges"`: the genomic positions of the regions of interest

`bin_number`**:** Object of class `"numeric"`: the number of bins per region

Methods

genome_name `signature(object = "MEDIPSroiSet")`: extracts the reference genome of the MEDIPS ROI SET

bin_number `signature(object = "MEDIPSroiSet")`: extracts the number of bins per ROI the bin_number slot of the MEDIPS ROI SET

chr_names `signature(object = "MEDIPSroiSet")`: extracts the names of the chromosomes included within the MEDIPS ROI SET

chr_lengths `signature(object = "MEDIPSroiSet")`: extracts the length of the chromosomes included within the MEDIPS ROI SET

sample_name `signature(object = "MEDIPSroiSet")`: extracts the name of the input file

path_name `signature(object = "MEDIPSroiSet")`: extracts the path to the input file

number_regions `signature(object = "MEDIPSroiSet")`: extracts the total number of included regions

genome_count `signature(object = "MEDIPSroiSet")`: extracts the raw MeDIP-Seq signals at the genomic bins

extend `signature(object = "MEDIPSroiSet")`: extracts the number of bases by which the regions are extended

show `signature(object = "MEDIPSroiSet")`: prints a summary of the MEDIPS SET object content

shifted `signature(object = "MEDIPSroiSet")`: extracts the number of bases by which the regions are shifted

uniq `signature(object = "MEDIPSroiSet")`: extracts the specified value for the uniq parameter

rois `signature(object = "MEDIPSroiSet")`: extracts the GRange object containing the regions of interest

MEDIPS.calibrationCurve `signature(MSet = "MEDIPSroiSet", CSet="COUPLINGset")`: internal function for calculating the calibration curve

MEDIPS.negBin `signature(MSet="MEDIPSroiSet", CSet="COUPLINGset")`: internal function for calculating methylatiopn probabilities with respect to CpG density dependent negative binomial distributions

MEDIPS.pois `signature(MSet="MEDIPSroiSet", CSet="COUPLINGset")`: internal function for calculating methylatiopn probabilities with respect to CpG density dependent poisson distributions

MEDIPS.rms `signature(MSet="MEDIPSroiSet", CSet="COUPLINGset")`: internal function for calculating relative methylation scores

Author(s)

Lukas Chavez, Joern Dietrich

Examples

```
showClass("MEDIPSroiSet")
```

`MEDIPSset-class` *MEDIPSset class and internal functions*

Description

MEDIPSset class is used in the MEDIPS library in order to store and extract objects and information from the input file as well as parameter settings specified during the workflow.

Objects from the Class

Objects of the classes contain information about the provided short reads, MeDIP raw/count signals, and further parameter settings. A MEDIPS SET object is created by the MEDIPS.genomeVector() function. According slots will be filled during the workflow.

Slots

`genome_name`: Object of class `"character"` : the refernce genome

`window_size`: Object of class `"numeric"` : the window size for the genome vector

`chr_names`: Object of class `"character"` : the names of the chromosomes included within the MEDIPS/COUPLING SET

`chr_lengths`: Object of class `"numeric"` : the lengths of the chromosomes included within the MEDIPS/COUPLING SET

`sample_name`: Object of class `"character"` : the name of the input file

`path_name`: Object of class `"character"` : the path to the input file

`number_regions`: Object of class `"numeric"` : the total number of included regions

`genome_count`: Object of class `"numeric"` : the raw MeDIP-seq signals at the genomic bins

`extend`: Object of class `"numeric"` : the length of the regions after extension

`shifted`: Object of class `"numeric"` : the number of bases by which the reads are shifted along the sequencing direction

`uniq`: Object of class `"logical"` : determines if reads mapping to exactly the same genomic position should be replaced by only one representative

Methods

genome_name `signature(object = "MEDIPSset")`: extracts the reference genome of the MEDIPS SET

window_size `signature(object = "MEDIPSset")`: extracts the window size from the bin_size slot of the MEDIPS SET

chr_names `signature(object = "MEDIPSset")`: extracts the names of the chromosomes included within the MEDIPS SET

chr_lengths `signature(object = "MEDIPSset")`: extracts the length of the chromosomes included within the MEDIPS SET

fragmentLength `signature(object = "MEDIPSset")`: extracts the estimated fragment length of the DNA fragments

sample_name `signature(object = "MEDIPSset")`: extracts the name of the input file

path_name `signature(object = "MEDIPSset")`: extracts the path to the input file

number_regions `signature(object = "MEDIPSset")`: extracts the total number of included regions

genome_count `signature(object = "MEDIPSset")`: extracts the raw MeDIP-Seq signals at the genomic bins

extend `signature(object = "MEDIPSset")`: extracts the number of bases by which the regions are extended

show `signature(object = "MEDIPSset")`: prints a summary of the MEDIPS SET object content

shifted `signature(object = "MEDIPSset")`: extracts the number of bases by which the regions are shifted

uniq `signature(object = "MEDIPSset")`: extracts the specified value for the uniq parameter

MEDIPS.distributeReads `signature(object = "MEDIPSset")`: internal function for distributing the reads over the genome vector

MEDIPS.GenomicCoordinates `signature(object = "MEDIPSset")`: internal function for calculating coordinates for the genomic bins

MEDIPS.readRegionsFile `signature(object = "MEDIPSset")`: internal function for reading short read information

MEDIPS.calibrationCurve `signature(object = "MEDIPSset")`: internal function for calculating the calibration curve

MEDIPS.cnv `signature(object = "MEDIPSset")`: internal function for calculating CNVs in case two groups of INPUT SETs have been provided to the MEDIPS.meth function

MEDIPS.diffMeth `signature(object = "MEDIPSset")`: internal function for calculating differential coverage in case two groups of MEDIPS SETs have been provided to the MEDIPS.meth function

MEDIPS.getPositions `signature(object = "MEDIPSset")`: internal function for receiving genomic coordinates of a given sequence pattern (e.g. CG)

MEDIPS.negBin `signature(object = "MEDIPSset")`: internal function for calculating methylatiopn probabilities with respect to CpG density dependent negative binomial distributions

MEDIPS.pois `signature(object = "MEDIPSset")`: internal function for calculating methylatiopn probabilities with respect to CpG density dependent poisson distributions

MEDIPS.rms `signature(object = "MEDIPSset")`: internal function for calculating relative methylation scores

matNnotNA `signature(object = "MEDIPSset")`: internal function for vectorized calculation of the t.test

matMin `signature(object = "MEDIPSset")`: internal function for vectorized calculation of the t.test

matDiff `signature(object = "MEDIPSset")`: internal function for vectorized calculation of the t.test

matMax `signature(object = "MEDIPSset")`: internal function for vectorized calculation of the t.test

matMean `signature(object = "MEDIPSset")`: internal function for vectorized calculation of the t.test

matSd `signature(object = "MEDIPSset")`: internal function for vectorized calculation of the t.test

matTtest `signature(object = "MEDIPSset")`: internal function for vectorized calculation of the t.test

Author(s)

Lukas Chavez, Joern Dietrich

Examples

```
showClass("MEDIPSset")
```

Index