

# Package ‘methrix’

October 7, 2024

**Title** Fast and efficient summarization of generic bedGraph files from Bisulfite sequencing

**Version** 1.19.0

**Description** Bedgraph files generated by Bisulfite pipelines often come in various flavors. Critical downstream step requires summarization of these files into methylation/coverage matrices. This step of data aggregation is done by Methrix, including many other useful downstream functions.

**License** MIT + file LICENSE

**Encoding** UTF-8

**LazyData** false

**Depends** R (>= 3.6), data.table (>= 1.12.4), SummarizedExperiment

**Imports** rtracklayer, DelayedArray, HDF5Array, BSgenome, DelayedMatrixStats, parallel, methods, ggplot2, S4Vectors, matrixStats, graphics, stats, utils, GenomicRanges, IRanges

**RoxygenNote** 7.1.1

**Suggests** knitr, rmarkdown, DSS, bsseq, plotly, BSgenome.Mmusculus.UCSC.mm9, MafDb.1Kgenomes.phase3.GRCh38, MafDb.1Kgenomes.phase3.hs37d5, BSgenome.Hsapiens.UCSC.hg19, GenomicScores, Biostrings, RColorBrewer, GenomeInfoDb, testthat (>= 2.1.0)

**VignetteBuilder** knitr

**biocViews** DNAMethylation, Sequencing, Coverage

**URL** <https://github.com/CompEpigen/methrix>

**BugReports** <https://github.com/CompEpigen/methrix/issues>

**git\_url** <https://git.bioconductor.org/packages/methrix>

**git\_branch** devel

**git\_last\_commit** 34add75

**git\_last\_commit\_date** 2024-04-30

**Repository** Bioconductor 3.20

**Date/Publication** 2024-10-06

**Author** Anand Mayakonda [aut, cre] (<<https://orcid.org/0000-0003-1162-687X>>),  
Reka Toth [aut] (<<https://orcid.org/0000-0002-6096-1052>>),  
Rajbir Batra [ctb],

Clarissa Feuerstein-Akgöz [ctb],  
 Joschka Hey [ctb],  
 Maximilian Schönung [ctb],  
 Pavlo Lutsik [ctb]

**Maintainer** Anand Mayakonda <anand\_mt@hotmail.com>

## Contents

combine_methrix . . . . .	2
convert_HDF5_methrix . . . . .	3
convert_methrix . . . . .	4
coverage_filter . . . . .	4
extract_CPGs . . . . .	5
get_matrix . . . . .	6
get_region_summary . . . . .	7
get_stats . . . . .	8
load_HDF5_methrix . . . . .	9
mask_methrix . . . . .	9
methrix-class . . . . .	10
methrix2bsseq . . . . .	11
methrix_data . . . . .	11
methrix_pca . . . . .	12
methrix_report . . . . .	13
order_by_sd . . . . .	14
plot_coverage . . . . .	14
plot_density . . . . .	15
plot_pca . . . . .	16
plot_stats . . . . .	17
plot_violin . . . . .	18
read_bedgraphs . . . . .	18
region_filter . . . . .	21
remove_snps . . . . .	21
remove_uncovered . . . . .	23
save_HDF5_methrix . . . . .	23
subset_methrix . . . . .	24
write_bedgraphs . . . . .	25
write_bigwigs . . . . .	26
<b>Index</b>	<b>27</b>

---

combine_methrix	<i>Combine methrix objects</i>
-----------------	--------------------------------

---

### Description

Combine methrix objects

### Usage

```
combine_methrix(m1, m2, by = c("row", "col"))
```

**Arguments**

m1	Frist <code>methrix</code> object
m2	Second <code>methrix</code> object
by	The direction of combine. 'column' (cbind) combines samples with same regions, 'row' combines different regions, e.g. different chromosomes.

**Details**

Takes two `methrix` objects and combines them row- or column-wise

**Value**

An object of class `methrix`

---

`convert_HDF5_methrix` *Converts HDF5 methrix object to standard in-memory object.*

---

**Description**

Converts HDF5 methrix object to standard in-memory object.

**Usage**

```
convert_HDF5_methrix(m = NULL)
```

**Arguments**

m An object of class `methrix`, HDF5 format

**Details**

Takes a `methrix` object and returns with the same object with in-memory assay slots.

**Value**

An object of class `methrix`

**Examples**

```
data(methrix_data)
m2 <- convert_methrix(m=methrix_data)
m <- convert_HDF5_methrix(m=m2)
```

---

convert_methrix	<i>Converts an in-memory object to an on-disk HDF5 object.</i>
-----------------	--

---

### Description

Converts an in-memory object to an on-disk HDF5 object.

### Usage

```
convert_methrix(m = NULL)
```

### Arguments

m                    An object of class `methrix`

### Details

Takes a `methrix` object and returns with the same object with delayed array assay slots with HDF5 backend. Might take long time!

### Value

An object of class `methrix`, HDF5 format

### Examples

```
data(methrix_data)
m2 <- convert_methrix(m=methrix_data)
```

---

coverage_filter	<i>Filter matrices by coverage</i>
-----------------	------------------------------------

---

### Description

Filter matrices by coverage

### Usage

```
coverage_filter(  
  m,  
  cov_thr = 1,  
  min_samples = 1,  
  prop_samples = 0,  
  group = NULL,  
  n_chunks = 1,  
  n_cores = 1  
)
```

**Arguments**

<code>m</code>	<code>methrix</code> object
<code>cov_thr</code>	minimum coverage required to call a loci covered
<code>min_samples</code>	Minimum number of samples that should have a loci with coverage $\geq$ <code>cov_thr</code> . If <code>group</code> is given, then this applies per group. Only need one of <code>prop_samples</code> or <code>min_samples</code> .
<code>prop_samples</code>	Minimum proportion of samples that should have a loci with coverage $\geq$ <code>cov_thr</code> . If <code>group</code> is given, then this applies per group. Only need one of <code>prop_samples</code> or <code>min_samples</code> .
<code>group</code>	a column name from sample annotation that defines groups. In this case, the number of <code>min_samples</code> will be tested group-wise.
<code>n_chunks</code>	Number of chunks to split the <code>methrix</code> object in case it is very large. Default = 1.
<code>n_cores</code>	Number of parallel instances. <code>n_cores</code> should be less than or equal to <code>n_chunks</code> . If <code>n_chunks</code> is not specified, then <code>n_chunks</code> is initialized to be equal to <code>n_cores</code> . Default = 1.

**Details**

Takes `methrix` object and filters CpGs based on coverage statistics

**Value**

An object of class `methrix`

**Examples**

```
data('methrix_data')
#keep only CpGs which are covered by at-least 1 read across 3 samples
coverage_filter(m = methrix_data, cov_thr = 1, min_samples = 3)
```

---

<code>extract_CPGs</code>	<i>Extracts all CpGs from a genome</i>
---------------------------	--

---

**Description**

Extracts all CpGs from a genome

**Usage**

```
extract_CPGs(ref_genome = NULL)
```

**Arguments**

`ref_genome` BSgenome object or name of the installed BSgenome package. Example: `BSgenome.Hsapiens.UCSC`

**Value**

a list of `data.table` containing number of CpG's and contig lengths

**Examples**

```
## Not run:
hg19_cpgs = methrix::extract_CPGs(ref_genome = 'BSgenome.Hsapiens.UCSC.hg19')

## End(Not run)
```

---

get\_matrix

*Extract methylation or coverage matrices*


---

**Description**

Extract methylation or coverage matrices

**Usage**

```
get_matrix(m, type = "M", add_loci = FALSE, in_granges = FALSE)
```

**Arguments**

m	<a href="#">methrix</a> object
type	can be M or C. Default 'M'
add_loci	Default FALSE. If TRUE adds CpG position info to the matrix and returns as a data.table
in_granges	Do you want the outcome in GRanges?

**Details**

Takes [methrix](#) object and returns user specified methylation or coverage matrix

**Value**

Coverage or Methylation matrix

**Examples**

```
data('methrix_data')
#Get methylation matrix
get_matrix(m = methrix_data, type = 'M')
#Get methylation matrix along with loci
get_matrix(m = methrix_data, type = 'M', add_loci = TRUE)
#' #Get methylation data as a GRanges object
get_matrix(m = methrix_data, type = 'M', add_loci = TRUE, in_granges=TRUE)
```

---

get_region_summary	<i>Extract and summarize methylation or coverage info by regions of interest</i>
--------------------	--

---

## Description

Extract and summarize methylation or coverage info by regions of interest

## Usage

```
get_region_summary(  
  m,  
  regions = NULL,  
  type = "M",  
  how = "mean",  
  overlap_type = "within",  
  na_rm = TRUE,  
  elementMetadata.col = NULL,  
  verbose = TRUE,  
  n_chunks = 1,  
  n_cores = 1  
)
```

## Arguments

m	<code>methrix</code> object
regions	genomic regions to be summarized. Could be a <code>data.table</code> with 3 columns (chr, start, end) or a <code>GenomicRanges</code> object
type	matrix which needs to be summarized. Could be 'M', 'C'. Default 'M'
how	mathematical function by which regions should be summarized. Can be one of the following: mean, sum, max, min. Default 'mean'
overlap_type	defines the type of the overlap of the CpG sites with the target region. Default value is 'within'. For detailed description, see the <code>findOverlaps</code> function of the <code>IRanges</code> package.
na_rm	Remove NA's? Default TRUE
elementMetadata.col	columns in <code>rowData(methrix)</code> which needs to be summarised. Default = NULL.
verbose	Default TRUE
n_chunks	Number of chunks to split the <code>methrix</code> object in case it is very large. Default = 1.
n_cores	Number of parallel instances. <code>n_cores</code> should be less than or equal to <code>n_chunks</code> . If <code>n_chunks</code> is not specified, then <code>n_chunks</code> is initialized to be equal to <code>n_cores</code> . Default = 1.

## Details

Takes `methrix` object and summarizes regions

**Value**

a coverage or methylation matrix

**Examples**

```
data('methrix_data')
get_region_summary(m = methrix_data,
regions = data.table(chr = 'chr21', start = 27867971, end = 27868103),
type = 'M', how = 'mean')
```

---

get\_stats

*Estimate descriptive statistics*

---

**Description**

Estimate descriptive statistics

**Usage**

```
get_stats(m, per_chr = TRUE)
```

**Arguments**

m [methrix](#) object  
per\_chr Estimate stats per chromosome. Default TRUE

**Details**

Calculate descriptive statistics

**Value**

data.table of summary stats

**See Also**

[plot\\_stats](#)

**Examples**

```
data('methrix_data')
get_stats(methrix_data)
```



---

load_HDF5_methrix	<i>Loads HDF5 methrix object</i>
-------------------	----------------------------------

---

### Description

Loads HDF5 methrix object

### Usage

```
load_HDF5_methrix(dir = NULL, ...)
```

### Arguments

dir	The directory to read in from. Default NULL
...	Parameters to pass to loadHDF5SummarizedExperiment

### Details

Takes directory with a previously saved HDF5Array format [methrix](#) object and loads it

### Value

An object of class [methrix](#)

### Examples

```
data('methrix_data')
methrix_data_h5 <- convert_methrix(m=methrix_data)
target_dir = paste0(getwd(), '/temp1/')
save_HDF5_methrix(methrix_data_h5, dir = target_dir, replace = TRUE)
load_HDF5_methrix(target_dir)
```

---

mask_methrix	<i>Masks too high or too low coverage</i>
--------------	---

---

### Description

Masks too high or too low coverage

### Usage

```
mask_methrix(m, low_count = NULL, high_quantile = 0.99, n_cores = 1)
```

**Arguments**

<code>m</code>	<code>methrix</code> object
<code>low_count</code>	The minimal coverage allowed. Everything below, will get masked. Default = NULL, nothing gets masked.
<code>high_quantile</code>	The quantile limit of coverage. Quantiles are calculated for each sample and everything that belongs to a higher quantile than the defined will be masked. Default = 0.99.
<code>n_cores</code>	Number of parallel instances. Can only be used if <code>methrix</code> is in HDF5 format. Default = 1.

**Details**

Takes `methrix` object and masks sites with too high or too low coverage by putting NA for coverage and beta value. The sites will remain in the object.

**Value**

An object of class `methrix`

**Examples**

```
data('methrix_data')
mask_methrix(m = methrix_data, low_count = 5, high_quantile = 0.99 )
```

---

methrix-class

*Class methrix*

---

**Description**

S4 class Methrix

**Slots**

`assays` A list of two matrices containing 'Methylation' and 'Coverage' information

`elementMetadata` A DataFrame describing rows in corresponding assay matrices.

`colData` genome: the name of the BSgenome that was used to extract CpGs, isHDF5: is it stored in HDF5 Array format

`metadata` a list of meta data associated with the assays

`NAMES` NULL

---

methrix2bsseq	Convert <code>methrix</code> to bsseq object
---------------	--

---

### Description

Convert `methrix` to bsseq object

### Usage

```
methrix2bsseq(m)
```

### Arguments

`m` `methrix` object

### Details

Takes `methrix` object and returns a bsseq object

### Value

An object of class bsseq

### Examples

```
## Not run:  
data('methrix_data')  
methrix2bsseq(m = methrix_data)  
  
## End(Not run)
```

---

methrix_data	<i>WGBS for colon cancer, chr21 and chr22</i>
--------------	---

---

### Description

This is a subset of original 'bsseqData' converted to 'methrix' containing Whole-genome bisulfite sequencing data (WGBS) for colon cancer on chromosome 21 and 22.

### Usage

```
data('methrix_data')
```

### Format

An object of class 'methrix'

### References

Hansen, K. D. et al. (2011) Increased methylation variation in epigenetic domains across cancer types. Nature Genetics 43, 768-775.

**Examples**

```
data('methrix_data')
methrix_data
```

---

methrix\_pca

*Principal Component Analysis*


---

**Description**

Principal Component Analysis

**Usage**

```
methrix_pca(
  m,
  var = "top",
  top_var = 1000,
  ranges = NULL,
  pheno = NULL,
  do_plot = TRUE,
  n_pc = 2
)
```

**Arguments**

m	Input <i>methrix</i> object
var	Choose between random CpG sites ('rand') or most variable CpGs ('top').
top_var	Number of variable CpGs to use. Default 1000 Set it to NULL to use all CpGs (which is not recommended due to memory requirements). This option is mutually exclusive with ranges.
ranges	genomic regions to be summarized. Could be a data.table with 3 columns (chr, start, end) or a GenomicRanges object
pheno	Column name of colData(m). Default NULL. Will be used as a factor to color different groups
do_plot	Should a plot be generated?
n_pc	Default 2.

**Value**

PCA results

**Examples**

```
data('methrix_data')
methrix_pca(methrix_data, do_plot = FALSE)
```

---

methrix_report	<i>Creates a detailed interactive html summary report from Methrix object</i>
----------------	---

---

### Description

Creates a detailed interactive html summary report from Methrix object. If the directory contains required files (from previous run), it directly proceeds to generate html report.

### Usage

```
methrix_report(  
  meth,  
  output_dir = NULL,  
  recal_stats = FALSE,  
  plot_beta_dist = TRUE,  
  beta_nCpG = 10000,  
  prefix = NULL,  
  n_thr = 4  
)
```

### Arguments

meth	<code>methrix</code> object
output_dir	Output directory name where the files should be saved. If NULL creates a tempdir
recal_stats	Whether summary statistics should be recalculated? If you are using subsetted methrix object set this to TRUE.
plot_beta_dist	Default TRUE. Can be time consuming.
beta_nCpG	Number of CpGs rto use for estimating beta value distribution. Default 10000
prefix	If provided, the name of the report and the intermediate files will start with the prefix.
n_thr	Default 4. Only used if plot_beta_dist is TRUE

### Value

an interactive html report

### Examples

```
## Not run:  
data('methrix_data')  
methrix::methrix_report(meth = methrix_data)  
  
## End(Not run)
```

---

order_by_sd	<i>Order methrix object by SD</i>
-------------	-----------------------------------

---

**Description**

Order methrix object by SD

**Usage**

```
order_by_sd(m)
```

**Arguments**

m                    [methrix](#) object

**Details**

Takes [methrix](#) object and reorganizes the data by standard deviation

**Value**

An object of class [methrix](#)

**Examples**

```
data('methrix_data')
order_by_sd(m = methrix_data)
```

---

plot_coverage	<i>Coverage QC Plots</i>
---------------	--------------------------

---

**Description**

Coverage QC Plots

**Usage**

```
plot_coverage(  
  m,  
  type = c("hist", "dens"),  
  pheno = NULL,  
  perGroup = FALSE,  
  lim = 100,  
  size.lim = 1e+06,  
  col_palette = "RdYlGn"  
)
```

**Arguments**

m	Input <code>methrix</code> object
type	Choose between 'hist' (histogram) or 'dens' (density plot).
pheno	Column name of <code>colData(m)</code> . Will be used as a factor to color different groups in the plot.
perGroup	Color the plots in a sample-wise manner?
lim	Maximum coverage value to be plotted.
size.lim	The maximum number of observations (sites*samples) to use. If the dataset is larger than this, random sites will be selected from the genome.
col_palette	Name of the RColorBrewer palette to use for plotting.

**Value**

ggplot2 object

**Examples**

```
data('methrix_data')
plot_coverage(m = methrix_data)
```

---

plot_density	<i>Density Plot of <math>\beta</math>-Values</i>
--------------	--

---

**Description**

Density Plot of  $\beta$ -Values

**Usage**

```
plot_density(
  m,
  ranges = NULL,
  n_cpgs = 25000,
  pheno = NULL,
  col_palette = "RdYlGn"
)
```

**Arguments**

m	Input <code>methrix</code> object
ranges	genomic regions to be summarized. Could be a <code>data.table</code> with 3 columns (chr, start, end) or a <code>GenomicRanges</code> object
n_cpgs	Use these many random CpGs for plotting. Default 25000. Set it to NULL to use all - which can be memory expensive.
pheno	Column name of <code>colData(m)</code> . Will be used as a factor to color different groups in the violin plot.
col_palette	Name of the RColorBrewer palette to use for plotting.

**Value**

ggplot2 object

**Examples**

```
data('methrix_data')
plot_density(m = methrix_data)
```

---

plot\_pca

*Plot PCA results*

---

**Description**

Plot PCA results

**Usage**

```
plot_pca(
  pca_res,
  m = NULL,
  col_anno = NULL,
  shape_anno = NULL,
  pc_x = "PC1",
  pc_y = "PC2",
  show_labels = FALSE
)
```

**Arguments**

pca_res	Results from <a href="#">methrix_pca</a>
m	optimal methrix object. Default NULL
col_anno	Column name of colData(m). Default NULL. Will be used as a factor to color different groups. Required methrix object
shape_anno	Column name of colData(m). Default NULL. Will be used as a factor to shape different groups. Required methrix object
pc_x	Default 'PC1'
pc_y	Default 'PC2'
show_labels	Default FLASE

**Value**

ggplot2 object

**Examples**

```
data('methrix_data')
mpc = methrix_pca(methrix_data, do_plot = FALSE)
plot_pca(mpc)
```



---

plot\_stats

*Plot descriptive statistics*

---

### Description

Plot descriptive statistics

### Usage

```
plot_stats(  
  plot_dat,  
  what = "M",  
  stat = "mean",  
  ignore_chr = NULL,  
  samples = NULL,  
  n_col = NULL,  
  n_row = NULL  
)
```

### Arguments

plot_dat	results from <a href="#">get_stats</a>
what	Can be M or C. Default M
stat	Can be mean or median. Default mean
ignore_chr	Chromosomes to ignore. Default NULL
samples	Use only these samples. Default NULL
n_col	number of columns. Passed to 'facet_wrap'
n_row	number of rows. Passed to 'facet_wrap'

### Details

plot descriptive statistics results from [get\\_stats](#)

### Value

ggplot2 object

### See Also

[get\\_stats](#)

### Examples

```
data('methrix_data')  
gs = get_stats(methrix_data)  
plot_stats(gs)
```

---

plot\_violin                      *Violin Plot for  $\beta$ -Values*

---

### Description

Violin Plot for  $\beta$ -Values

### Usage

```
plot_violin(  
  m,  
  ranges = NULL,  
  n_cpgs = 25000,  
  pheno = NULL,  
  col_palette = "RdYlGn"  
)
```

### Arguments

m	Input <code>methrix</code> object
ranges	genomic regions to be summarized. Could be a <code>data.table</code> with 3 columns (chr, start, end) or a <code>GenomicRanges</code> object
n_cpgs	Use these many random CpGs for plotting. Default 25000. Set it to NULL to use all - which can be memory expensive.
pheno	Column name of <code>colData(m)</code> . Will be used as a factor to color different groups in the violin plot.
col_palette	Name of the <code>RColorBrewer</code> palette to use for plotting.

### Value

ggplot2 object

### Examples

```
data('methrix_data')  
plot_violin(m = methrix_data)
```

---

read\_bedgraphs                      *Versatile BedGraph reader.*

---

### Description

Versatile BedGraph reader.

**Usage**

```

read_bedgraphs(
  files = NULL,
  pipeline = NULL,
  zero_based = TRUE,
  stranded = FALSE,
  collapse_strands = FALSE,
  ref_cpgs = NULL,
  ref_build = NULL,
  contigs = NULL,
  vect = FALSE,
  vect_batch_size = NULL,
  coldata = NULL,
  chr_idx = NULL,
  start_idx = NULL,
  end_idx = NULL,
  beta_idx = NULL,
  M_idx = NULL,
  U_idx = NULL,
  strand_idx = NULL,
  cov_idx = NULL,
  synced_coordinates = FALSE,
  n_threads = 1,
  h5 = FALSE,
  h5_dir = NULL,
  h5temp = NULL,
  verbose = TRUE
)

```

**Arguments**

files	bedgraph files.
pipeline	Default NULL. Currently supports "Bismark_cov", "MethylDackel", "MethylcTools", "BisSNP", "BSseeker2_CGmap" If not known use idx arguments for manual column assignments.
zero_based	Are bedgraph regions zero based ? Default TRUE
stranded	Default FALSE
collapse_strands	If TRUE collapses CpGs on different crick strand into watson. Deafult FALSE
ref_cpgs	BSgenome object, or name of the installed BSgenome package, or an output from <a href="#">extract_CPGs</a> . Example: BSgenome.Hsapiens.UCSC.hg19
ref_build	reference genome for bedgraphs. Default NULL. Only used for additional details. Doesnt affect in any way.
contigs	contigs to restrict genomic CpGs to. Default all autosomes and allosomes - ignoring extra contigs.
vect	To use vectorized code. Default FALSE. Set to TRUE if you don't have large number of BedGraph files.
vect_batch_size	Default NULL. Process samples in batches. Applicable only when vect = TRUE

coldata	An optional DataFrame describing the samples. Row names, if present, become the column names of the matrix. If NULL, then a DataFrame will be created with basename of files used as the row names.
chr_idx	column index for chromosome in bedgraph files
start_idx	column index for start position in bedgraph files
end_idx	column index for end position in bedgraph files
beta_idx	column index for beta values in bedgraph files
M_idx	column index for read counts supporting Methylation in bedgraph files
U_idx	column index for read counts supporting Un-methylation in bedgraph files
strand_idx	column index for strand information in bedgraph files
cov_idx	column index for total-coverage in bedgraph files
synced_coordinates	Are the start and end coordinates of a stranded bedgraph are synchronized between + and - strands? Possible values: FALSE (default), TRUE if the start coordinates are the start coordinates of the C on the plus strand.
n_threads	number of threads to use. Default 1. Be-careful - there is a linear increase in memory usage with number of threads. This option is does not work with Windows OS.
h5	Should the coverage and methylation matrices be stored as 'HDF5Array'
h5_dir	directory to store H5 based object
h5temp	temporary directory to store hdf5
verbose	Be little chatty ? Default TRUE.

## Details

Reads BedGraph files and generates methylation and coverage matrices. Optionally arrays can be serialized as on-disk HDFS5 arrays.

## Value

An object of class `methrix`

## Examples

```
## Not run:
bdg_files = list.files(path = system.file('extdata', package = 'methrix'),
pattern = '*\\.bedGraph\\.gz$', full.names = TRUE)
hg19_cpgs = methrix::extract_CPGs(ref_genome = 'BSgenome.Hsapiens.UCSC.hg19')
meth = methrix::read_bedgraphs( files = bdg_files, ref_cpgs = hg19_cpgs,
chr_idx = 1, start_idx = 2, M_idx = 3, U_idx = 4,
stranded = FALSE, zero_based = FALSE, collapse_strands = FALSE)

## End(Not run)
```

---

region_filter	<i>Filter matrices by region</i>
---------------	----------------------------------

---

### Description

Filter matrices by region

### Usage

```
region_filter(m, regions, type = "within")
```

### Arguments

m	<a href="#">methrix</a> object
regions	genomic regions to filter-out. Could be a <code>data.table</code> with 3 columns (chr, start, end) or a <code>GenomicRanges</code> object
type	defines the type of the overlap of the CpG sites with the target regions. Default value is 'within'. For detailed description, see the <code>foverlaps</code> function of the <a href="#">data.table</a> package.

### Details

Takes [methrix](#) object and filters CpGs based on supplied regions in `data.table` or `GRanges` format

### Value

An object of class [methrix](#)

### Examples

```
data('methrix_data')
region_filter(m = methrix_data,
regions = data.table(chr = 'chr21', start = 27867971, end = 27868103))
```

---

remove_snps	<i>Removes CpG sites from the object if they overlap with common SNPs</i>
-------------	---

---

### Description

Removes CpG sites from the object if they overlap with common SNPs

**Usage**

```
remove_snps(
  m,
  populations = NULL,
  maf_threshold = 0.01,
  reduce_filtering = FALSE,
  forced = FALSE,
  keep = FALSE,
  n_chunks = 1,
  n_cores = 1
)
```

**Arguments**

<code>m</code>	<code>methrix</code> object
<code>populations</code>	Populations to use. Default is all.
<code>maf_threshold</code>	The frequency threshold, above which the SNPs will be removed. Default is 0.01
<code>reduce_filtering</code>	If TRUE, the SNPs with a MAF < 0.1 will be evaluated and only the highly variable ones will be removed. Default FALSE.
<code>forced</code>	the <code>reduce_filtering</code> is not recommended with less than 10 samples, but can be forced. Default is FALSE.
<code>keep</code>	Do you want to keep the sites that were filtered out? In this case, the function will return with a list of two <code>methrix</code> objects.
<code>n_chunks</code>	Number of chunks to split the <code>methrix</code> object in case it is very large. Can only be used if input data is in HDF5 format. Default = 1.
<code>n_cores</code>	Number of parallel instances. Can only be used if input data is in HDF5 format. <code>n_cores</code> should be less than or equal to <code>n_chunks</code> . If <code>n_chunks</code> is not specified, then <code>n_chunks</code> is initialized to be equal to <code>n_cores</code> . Default = 1.

**Details**

Takes `methrix` object and removes common SNPs. SNPs overlapping with a CpG site and have a minor allele frequency (MAF) above a threshold in any of the populations used will be selected and the corresponding CpG sites will be removed from the `methrix` object. With the `reduce_filtering` option, SNPs with MAF < 0.1 will be further evaluated. If they show low variance in the dataset, there is probably no genotype variability in the population, therefore the corresponding CpG site won't be removed. Please keep in mind that variance thresholds are

**Value**

`methrix` object or a list of `methrix` objects

**Examples**

```
data('methrix_data')
remove_snps(m = methrix_data, maf_threshold=0.01)
```

---

remove_uncovered	<i>Remove loci that are uncovered across all samples</i>
------------------	--

---

**Description**

Remove loci that are uncovered across all samples

**Usage**

```
remove_uncovered(m)
```

**Arguments**

m                    [methrix](#) object

**Details**

Takes [methrix](#) object and removes loci that are uncovered across all samples

**Value**

An object of class [methrix](#)

**Examples**

```
data('methrix_data')
remove_uncovered(m = methrix_data)
```

---

save_HDF5_methrix	<i>Saves HDF5 methrix object</i>
-------------------	----------------------------------

---

**Description**

Saves HDF5 methrix object

**Usage**

```
save_HDF5_methrix(m = NULL, dir = NULL, replace = FALSE, ...)
```

**Arguments**

m                    [methrix](#) object  
dir                   The directory to use. Created, if not existing. Default NULL  
replace               Should it overwrite the pre-existing data? FALSE by default.  
...                   Parameters to pass to saveHDF5SummarizedExperiment

**Details**

Takes [methrix](#) object and saves it

**Value**

Nothing

**Examples**

```
data('methrix_data')
methrix_data_h5 <- convert_methrix(m=methrix_data)
target_dir = paste0(getwd(), '/temp/')
save_HDF5_methrix(methrix_data_h5, dir = target_dir, replace = TRUE)
```

---

subset\_methrix

*Subsets methrix object based on given conditions.*

---

**Description**

Subsets `methrix` object based on given conditions.

**Usage**

```
subset_methrix(
  m,
  regions = NULL,
  contigs = NULL,
  samples = NULL,
  overlap_type = "within"
)
```

**Arguments**

<code>m</code>	<code>methrix</code> object
<code>regions</code>	genomic regions to subset by. Could be a <code>data.table</code> with 3 columns (chr, start, end) or a <code>GenomicRanges</code> object
<code>contigs</code>	chromosome names to subset by
<code>samples</code>	sample names to subset by
<code>overlap_type</code>	defines the type of the overlap of the CpG sites with the target region. Default value is 'within'. For detailed description, see the <code>foverlaps</code> function of the <code>data.table</code> package.

**Details**

Takes `methrix` object and filters CpGs based on coverage statistics

**Value**

An object of class `methrix`

**Examples**

```
data('methrix_data')
#Subset to chromosome 1
subset_methrix(methrix_data, contigs = 'chr21')
```



---

write_bedgraphs	<i>Writes bedGraphs from methrix object</i>
-----------------	---

---

### Description

Writes bedGraphs from methrix object

### Usage

```
write_bedgraphs(  
  m,  
  output_dir = NULL,  
  rm_NA = TRUE,  
  force = FALSE,  
  n_thr = 4,  
  compress = TRUE,  
  SeqStyle = "UCSC",  
  multiBed = NULL,  
  metilene = FALSE,  
  phenoCol = NULL,  
  add_coverage = FALSE  
)
```

### Arguments

m	<a href="#">methrix</a> object
output_dir	Output directory name where the files should be saved. If NULL creates a tempdir
rm_NA	remove NAs
force	forces to create files if they are existing
n_thr	Default 4.
compress	Whether to compress the output. Default TRUE
SeqStyle	Default 'UCSC' with 'chr' prefix.
multiBed	Default NULL. If provided a filename, a single bedGraph file with all samples included is generated.
metilene	Default FALSE. If TRUE outputs bedgraphs in 'metilene' format that can be directly used for DMR calling with 'metilene'. This option works only when multiBed = TRUE.
phenoCol	Default NULL. 'condition' column from colData. Only applicable if metilene = TRUE
add_coverage	Should the output file contain information on coverage? Default FALSE

### Value

writes bedgraph files to output

## Examples

```
data('methrix_data')
write_bedgraphs(m = methrix_data, output_dir = './temp')
#Export to metline format for DMR calling with metline
write_bedgraphs(m = methrix_data, output_dir = "./temp", rm_NA = FALSE,
metilene = TRUE,multiBed = "metline_ip", phenoCol = "Condition")
```

---

write_bigwigs	<i>Exports methrix object as bigWigs</i>
---------------	--

---

## Description

Exports methrix object as bigWigs

## Usage

```
write_bigwigs(m, output_dir = getwd(), samp_names = NULL)
```

## Arguments

m	<a href="#">methrix</a> object
output_dir	Output directory name where the files should be saved. Default getwd()
samp_names	sample names to export

## Examples

```
data('methrix_data')
write_bigwigs(m = methrix_data, output_dir = './temp')
```

# Index

## \* datasets

- methrix\_data, 11
  
- combine\_methrix, 2
- convert\_HDF5\_methrix, 3
- convert\_methrix, 4
- coverage\_filter, 4
  
- data.table, 21, 24
  
- extract\_CPGs, 5, 19
  
- get\_matrix, 6
- get\_region\_summary, 7
- get\_stats, 8, 17
  
- IRanges, 7
  
- load\_HDF5\_methrix, 9
  
- mask\_methrix, 9
- methrix, 3–15, 18, 20–26
- methrix (methrix-class), 10
- methrix-class, 10
- methrix2bsseq, 11
- methrix\_data, 11
- methrix\_pca, 12, 16
- methrix\_report, 13
  
- order\_by\_sd, 14
  
- plot\_coverage, 14
- plot\_density, 15
- plot\_pca, 16
- plot\_stats, 8, 17
- plot\_violin, 18
  
- read\_bedgraphs, 18
- region\_filter, 21
- remove\_snps, 21
- remove\_uncovered, 23
  
- save\_HDF5\_methrix, 23
- subset\_methrix, 24
  
- write\_bedgraphs, 25
- write\_bigwigs, 26