

# Package ‘methyISig’

July 17, 2025

**Title** MethyISig: Differential Methylation Testing for WGBS and RRBS Data

**Version** 1.21.0

**Date** 2020-04-22

**Description** MethyISig is a package for testing for differentially methylated cytosines (DMCs) or regions (DMRs) in whole-genome bisulfite sequencing (WGBS) or reduced representation bisulfite sequencing (RRBS) experiments. MethyISig uses a beta binomial model to test for significant differences between groups of samples. Several options exist for either site-specific or sliding window tests, and variance estimation.

**Depends** R (>= 3.6)

**Imports** bsseq, DelayedArray, DelayedMatrixStats, DSS, IRanges, GenomeInfoDb, GenomicRanges, methods, parallel, stats, S4Vectors

**Suggests** BiocStyle, bsseqData, knitr, rmarkdown, testthat (>= 2.1.0), covr

**License** GPL-3

**BugReports** <https://github.com/sartorlab/methyISig/issues>

**biocViews** DNAMethylation, DifferentialMethylation, Epigenetics, Regression, MethylSeq

**Encoding** UTF-8

**LazyData** true

**VignetteBuilder** knitr

**RoxygenNote** 7.1.0

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

**git\_branch** devel

**git\_last\_commit** ecbe132

**git\_last\_commit\_date** 2025-04-15

**Repository** Bioconductor 3.22

**Date/Publication** 2025-07-16

**Author** Yongseok Park [aut],  
Raymond G. Cavalcante [aut, cre]

**Maintainer** Raymond G. Cavalcante <[rcavalca@umich.edu](mailto:rcavalca@umich.edu)>

## Contents

bsseq_destranded . . . . .	2
bsseq_multichrom . . . . .	3
bsseq_stranded . . . . .	3
diff_binomial . . . . .	4
diff_dss_fit . . . . .	5
diff_dss_test . . . . .	6
diff_methylsig . . . . .	8
filter_loci_by_coverage . . . . .	9
filter_loci_by_group_coverage . . . . .	10
filter_loci_by_location . . . . .	11
methylSig . . . . .	12
promoters_gr . . . . .	12
tile_by_regions . . . . .	13
tile_by_windows . . . . .	13

<b>Index</b>	<b>15</b>
--------------	-----------

---

bsseq_destranded	<i>BSseq object read from destranded coverage files</i>
------------------	---------------------------------------------------------

---

## Description

Data contains 6 methylation loci and 2 samples

## Usage

```
bsseq_destranded
```

## Format

A BSseq object

## Source

```
data-raw/02-create_bsseq_rda.R
```

## Examples

```
data(bsseq_destranded, package = 'methylSig')
```

---

`bsseq_multichrom`*BSseq object with loci on multiple chromosomes*

---

**Description**

Data contains 4 methylation loci for 2 samples on 2 chromosomes

**Usage**

```
bsseq_multichrom
```

**Format**

A BSseq object

**Source**

```
data-raw/02-create_bsseq_rda.R
```

**Examples**

```
data(bsseq_multichrom, package = 'methylSig')
```

---

`bsseq_stranded`*BSseq object read from stranded coverage files*

---

**Description**

Data contains 11 methylation loci and 2 samples

**Usage**

```
bsseq_stranded
```

**Format**

A BSseq object

**Source**

```
data-raw/02-create_bsseq_rda.R
```

**Examples**

```
data(bsseq_stranded, package = 'methylSig')
```

diff\_binomial

*Differential methylation analysis using binomial model***Description**

This function calculates differential methylation statistics using a binomial-based approach. See ‘Warning’ message below.

**Usage**

```
diff_binomial(bs, group_column, comparison_groups)
```

**Arguments**

**bs** A BSseq-class object to calculate differential methylation statistics. See methylSigReadData for how to read in methylation data.

**group\_column** a character string indicating the column of pData(bs) to use for determining group membership.

**comparison\_groups** a named character vector indicating the case and control factors of group\_column for the comparison.

**Details**

This function uses a binomial-based model to calculate differential methylation statistics. It is nearly identical to the methylKit::calculateDiffMeth function in the methylKit R package except that only the likelihood ratio test and p.adjust(..., method='BH') are used to calculate significance levels. It is significantly faster than methylKit::calculateDiffMeth function.

**Value**

A GRanges object containing the following mcols:

**meth\_case:** Methylation estimate for case.

**meth\_control:** Methylation estimate for control.

**meth\_diff:** The difference meth\_case - meth\_control.

**direction:** The group for which the locus is hyper-methylated. Note, this is not subject to significance thresholds.

**pvalue:** The p-value from the t-test (t\_approx = TRUE) or the Chi-Square test (t\_approx = FALSE).

**fdr:** The Benjamini-Hochberg adjusted p-values using p.adjust(method = 'BH').

**log\_lik\_ratio:** The log likelihood ratio.

**Warning**

This function does not take into account the variability among samples in each group being compared.

## Examples

```
data(BS.cancer.ex, package = 'bsseqData')

bs = filter_loci_by_group_coverage(
  bs = BS.cancer.ex,
  group_column = 'Type',
  c('cancer' = 2, 'normal' = 2))

small_test = bs[1:50]

diff_gr = diff_binomial(
  bs = small_test,
  group_column = 'Type',
  comparison_groups = c('case' = 'cancer', 'control' = 'normal'))
```

---

diff_dss_fit	<i>Performs model fit for general experimental design</i>
--------------	-----------------------------------------------------------

---

## Description

This function is a wrapper for `DSS::DMLfit.multiFactor`.

## Usage

```
diff_dss_fit(bs, design, formula)
```

## Arguments

<b>bs</b>	a BSseq object to calculate differential methylation statistics.
<b>design</b>	a <code>data.frame</code> or <code>DataFrame</code> for experimental design. Should contain as many rows as there are columns (samples) in <code>bs</code> , and the order of the rows should match the columns of <code>bs</code> . If omitted, will default to <code>pData(bs)</code> .
<b>formula</b>	a formula for the linear model. It should refer to column names from <code>design</code> . NOTE: The intercept is included by default if omitted. One can omit the intercept with a formula such as <code>'~ 0 + group'</code> . For clarity, it helps to include the intercept explicitly as in <code>'~ 1 + group'</code> .

## Value

A list object with:

**gr:** a `GRanges` object with loci fit.

**design:** the `data.frame` input as the experimental design.

**formula:** the formula representing the model. Can be character or formula.

**X:** the design matrix used in regression based on the design and formula. This should be consulted to determine the appropriate contrast to use in `dss_fit_test()`.

**fit:** a list with model fitting results. It has components `beta`, the estimated coefficients, and `var.beta` the estimated variance/covariance matrix for `beta`.

## Examples

```
data(BS.cancer.ex, package = 'bsseqData')

bs = filter_loci_by_group_coverage(
  bs = BS.cancer.ex,
  group_column = 'Type',
  c('cancer' = 2, 'normal' = 2))

small_test = bs[1:50]

diff_fit = diff_dss_fit(
  bs = small_test,
  design = bsseq::pData(bs),
  formula = '~ Type')
```

---

diff_dss_test	<i>Calculates differential methylation statistics under general experimental design</i>
---------------	-----------------------------------------------------------------------------------------

---

## Description

This function is a wrapper for `DSS::DMLtest.multiFactor` with the added feature of reporting methylation rates alongside the test results via the `methylation_group_column` and `methylation_groups` parameters. See documentation below.

## Usage

```
diff_dss_test(
  bs,
  diff_fit,
  contrast,
  methylation_group_column = NA,
  methylation_groups = NA
)
```

## Arguments

<code>bs</code>	a BSseq, the same used used to create <code>diff_fit</code> .
<code>diff_fit</code>	a list object output by <code>diff_dss_fit()</code> .
<code>contrast</code>	a contrast matrix for hypothesis testing. The number of rows should match the number of columns design. Consult <code>diff_fit\$X</code> to ensure the contrast corresponds to the intended test.
<code>methylation_group_column</code>	Optionally, a column from <code>diff_fit\$design</code> by which to group samples and capture methylation rates. This column can be a character, factor, or numeric. In the case of numeric the samples are grouped according to the top and bottom 25 percentiles of the covariate, and the mean methylation for each group is calculated. If not a numeric, use the <code>methylation_groups</code> parameter to specify case and control.

methylation\_groups

Optionally, a named character vector indicating the case and control factors of methylation\_group\_column by which to group samples and capture methylation rates. If specified, must also specify methylation\_group\_column.

## Value

A GRanges object containing the following mcols:

**stat:** The test statistic.

**pvalue:** The p-value.

**fdr:** The Benjamini-Hochberg adjusted p-values using `p.adjust(method = 'BH')`.

If methylation\_group\_column is specified, also the following mcols:

**meth\_case:** Methylation estimate for case.

**meth\_control:** Methylation estimate for control.

**meth\_diff:** The difference `meth_case - meth_control`.

**direction:** The group for which the locus is hyper-methylated. Note, this is not subject to significance thresholds.

## Examples

```
data(BS.cancer.ex, package = 'bsseqData')

bs = filter_loci_by_group_coverage(
  bs = BS.cancer.ex,
  group_column = 'Type',
  c('cancer' = 2, 'normal' = 2))

small_test = bs[1:50]

diff_fit = diff_dss_fit(
  bs = small_test,
  design = bsseq::pData(bs),
  formula = '~ Type')

result = diff_dss_test(
  bs = small_test,
  diff_fit = diff_fit,
  contrast = matrix(c(0,1), ncol = 1)
)

result_with_meth = diff_dss_test(
  bs = small_test,
  diff_fit = diff_fit,
  contrast = matrix(c(0,1), ncol = 1),
  methylation_group_column = 'Type',
  methylation_groups = c('case' = 'cancer', 'control' = 'normal')
)
```

---

diff_methylsig	<i>Calculates differential methylation statistics using a Beta-binomial approach</i>
----------------	--------------------------------------------------------------------------------------

---

### Description

The function calculates differential methylation statistics between two groups of samples using a beta-binomial approach to calculate differential methylation statistics, accounting for variation among samples within each group. The function can be applied to a BSseq object subjected to `filter_loci_by_coverage()`, `filter_loci_by_snps()`, `filter_loci_by_group_coverage()` or any combination thereof. Moreover, the function can be applied to a BSseq object which has been tiled with `tile_by_regions()` or `tile_by_windows()`.

### Usage

```
diff_methylsig(
  bs,
  group_column,
  comparison_groups,
  disp_groups,
  local_window_size = 0,
  local_weight_function,
  t_approx = TRUE,
  n_cores = 1
)
```

### Arguments

<code>bs</code>	a BSseq object.
<code>group_column</code>	a character string indicating the column of <code>pData(bs)</code> to use for determining group membership.
<code>comparison_groups</code>	a named character vector indicating the case and control factors of <code>group_column</code> for the comparison.
<code>disp_groups</code>	a named logical vector indicating the whether to use case, control, or both to estimate the dispersion.
<code>local_window_size</code>	an integer indicating the size of the window for use in determining local information to improve mean and dispersion parameter estimations. In addition to a the distance constraint, a maximum of 5 loci upstream and downstream of the locus are used. The default is 0, indicating no local information is used.
<code>local_weight_function</code>	a weight kernel function. The default is the tri-weight kernel function defined as $\text{function}(u) = (1-u^2)^3$ . The domain of any given weight function should be $[-1,1]$ , and the range should be $[0,1]$ .
<code>t_approx</code>	a logical value indicating whether to use squared t approximation for the likelihood ratio statistics. Chi-square approximation ( <code>t_approx = FALSE</code> ) is recommended when the sample size is large. Default is TRUE.
<code>n_cores</code>	an integer denoting how many cores should be used for differential methylation calculations.



**Value**

A GRanges object containing the following mcols:

**meth\_case:** Methylation estimate for case.

**meth\_control:** Methylation estimate for control.

**meth\_diff:** The difference `meth_case - meth_control`.

**direction:** The group for which the locus is hyper-methylated. Note, this is not subject to significance thresholds.

**pvalue:** The p-value from the t-test (`t_approx = TRUE`) or the Chi-Square test (`t_approx = FALSE`).

**fdr:** The Benjamini-Hochberg adjusted p-values using `p.adjust(method = 'BH')`.

**disp\_est:** The dispersion estimate.

**log\_lik\_ratio:** The log likelihood ratio.

**df:** Degrees of freedom used when `t_approx = TRUE`.

**Examples**

```
data(BS.cancer.ex, package = 'bsseqData')

bs = filter_loci_by_group_coverage(
  bs = BS.cancer.ex,
  group_column = 'Type',
  c('cancer' = 2, 'normal' = 2))

small_test = bs[seq(50)]

diff_gr = diff_methylsig(
  bs = small_test,
  group_column = 'Type',
  comparison_groups = c('case' = 'cancer', 'control' = 'normal'),
  disp_groups = c('case' = TRUE, 'control' = TRUE),
  local_window_size = 0,
  t_approx = TRUE,
  n_cores = 1)
```

---

`filter_loci_by_coverage`

*Filter BSseq object by coverage*

---

**Description**

Used after `bsseq::read.bismark` to mark loci in samples below `min_count` or above `max_count` to 0. These loci will then be removed prior to differential analysis by `filter_loci_by_group_coverage()` if there are not a sufficient number of samples with appropriate coverage.

**Usage**

```
filter_loci_by_coverage(bs, min_count = 5, max_count = 500)
```

**Arguments**

bs	a BSseq object resulting from <code>bsseq::read.bismark</code> or constructed manually by the user.
min_count	an integer giving the minimum coverage required at a locus.
max_count	an integer giving the maximum coverage allowed at a locus.

**Value**

A BSseq object with samples/loci in the coverage and methylation matrix set to 0 where the coverage was less than `min_count` or greater than `max_count`. The number of samples and loci are conserved.

**Examples**

```
bis_cov_file1 = system.file('extdata', 'bis_cov1.cov', package = 'methylSig')
bis_cov_file2 = system.file('extdata', 'bis_cov2.cov', package = 'methylSig')
test = bsseq::read.bismark(
  files = c(bis_cov_file1, bis_cov_file2),
  colData = data.frame(row.names = c('test1', 'test2')),
  rmZeroCov = FALSE,
  strandCollapse = FALSE
)
test = filter_loci_by_coverage(bs = test, min_count = 10, max_count = 500)
```

---

`filter_loci_by_group_coverage`

*Filter loci based on coverage threshold per sample per group*

---

**Description**

An optional function to remove loci not satisfying coverage thresholds from `filter_loci_by_coverage` in a minimum number of samples per group.

**Usage**

```
filter_loci_by_group_coverage(bs, group_column, min_samples_per_group)
```

**Arguments**

bs	a BSseq object.
group_column	a character string indicating the column of <code>pData(bs)</code> to use for determining group membership.
min_samples_per_group	a named integer vector indicating the minimum number of samples with non-zero coverage required for maintaining a locus.

**Details**

The `filter_loci_by_coverage` function marked locus/sample pairs in the coverage matrix as 0 if said pair had coverage less than `minCount` or more than `maxCount`. This function enforces a threshold on the minimum number of samples per group required for a locus to be tested in downstream testing functions.

**Value**

A BSseq object with only those loci having min\_samples\_per\_group.

**Examples**

```
data(BS.cancer.ex, package = 'bsseqData')

filter_loci_by_group_coverage(
  bs = BS.cancer.ex,
  group_column = 'Type',
  min_samples_per_group = c('cancer' = 3, 'normal' = 3)
)
```

---

filter\_loci\_by\_location

*Remove loci by overlap with a GRanges object*

---

**Description**

A function to remove loci from a BSseq object based on intersection with loci in a GRanges object.

**Usage**

```
filter_loci_by_location(bs, gr)
```

**Arguments**

bs	a BSseq object.
gr	a GRanges object.

**Value**

A BSseq object with loci intersecting gr removed.

**Examples**

```
data(bsseq_stranded, package = 'methylSig')
regions = GenomicRanges::GRanges(
  seqnames = c('chr1', 'chr1', 'chr1', 'chr1'),
  ranges = IRanges::IRanges(
    start = c(5, 25, 45, 70),
    end = c(15, 40, 55, 80)
  )
)
filtered = filter_loci_by_location(bs = bsseq_stranded, gr = regions)
```

---

methyISig	<i>MethyISig: Differential Methylation Testing for WGBS and RRBS Data</i>
-----------	---------------------------------------------------------------------------

---

### Description

MethyISig is a package for testing for differentially methylated cytosines (DMCs) or regions (DMRs) in whole-genome bisulfite sequencing (WGBS) or reduced representation bisulfite sequencing (RRBS) experiments. MethyISig uses a beta binomial model to test for significant differences between groups of samples. Several options exist for either site-specific or sliding window tests, and variance estimation.

### methyISig functions

filter\_loci\_by\_coverage() filter\_loci\_by\_snps() tile\_by\_regions() tile\_by\_windows() filter\_loci\_by\_group\_coverage() diff\_binomial() diff\_methyISig() diff\_methyISig\_dss() annotate\_diff() visualize\_diff() region\_enrichment\_diff()

### Author(s)

**Maintainer:** Raymond G. Cavalcante <rcavalca@umich.edu>

Authors:

- Yongseok Park <yongpark@pitt.edu>

### See Also

Useful links:

- Report bugs at <https://github.com/sartorlab/methyISig/issues>

---

promoters_gr	<i>GRanges object with collapsed promoters on chr21 and chr22</i>
--------------	-------------------------------------------------------------------

---

### Description

Data contains 1466 promoters for use in the vignette

### Usage

```
promoters_gr
```

### Format

A GRanges object

### Source

data-raw/02-create\_bsseq\_rda.R

### Examples

```
data(promoters_gr, package = 'methyISig')
```

---

tile_by_regions	<i>Group cytosine / CpG level data into regions based on genomic regions</i>
-----------------	------------------------------------------------------------------------------

---

**Description**

An optional function to aggregate cytosine / CpG level data into regions based on a GRanges set of genomic regions.

**Usage**

```
tile_by_regions(bs, gr)
```

**Arguments**

bs	a BSseq object.
gr	a GRanges object.

**Value**

A BSseq object with loci of regions matching gr. Coverage and methylation read count matrices are aggregated by the sums of the cytosines / CpGs in the regions per sample.

**Examples**

```
data(bsseq_stranded, package = 'methySig')
regions = GenomicRanges::GRanges(
  seqnames = c('chr1', 'chr1', 'chr1'),
  ranges = IRanges::IRanges(
    start = c(5, 35, 75),
    end = c(30, 70, 80)
  )
)
tiled = tile_by_regions(bs = bsseq_stranded, gr = regions)
```

---

tile_by_windows	<i>Group cytosine / CpG level data into regions based on genomic windows</i>
-----------------	------------------------------------------------------------------------------

---

**Description**

An optional function to aggregate cytosine / CpG level data into regions based on a tiling of the genome by win\_size.

**Usage**

```
tile_by_windows(bs, win_size = 200)
```

**Arguments**

`bs` a BSseq object.  
`win_size` an integer indicating the size of the tiles. Default is 200bp.

**Value**

A BSseq object with loci consisting of a tiling of the genome by `win_size` bp tiles. Coverage and methylation read count matrices are aggregated by the sums of the cytosines / CpGs in the regions per sample.

**Examples**

```
data(bsseq_stranded, package = 'methylSig')  
  
tiled = tile_by_windows(bs = bsseq_stranded, win_size = 50)
```

# Index

## \* **datasets**

- bsseq\_destranded, [2](#)
- bsseq\_multichrom, [3](#)
- bsseq\_stranded, [3](#)
- promoters\_gr, [12](#)

## \* **internal**

- methyISig, [12](#)

bsseq\_destranded, [2](#)

bsseq\_multichrom, [3](#)

bsseq\_stranded, [3](#)

diff\_binomial, [4](#)

diff\_dss\_fit, [5](#)

diff\_dss\_test, [6](#)

diff\_methyISig, [8](#)

filter\_loci\_by\_coverage, [9](#)

filter\_loci\_by\_group\_coverage, [10](#)

filter\_loci\_by\_location, [11](#)

methyISig, [12](#)

methyISig-package (methyISig), [12](#)

promoters\_gr, [12](#)

tile\_by\_regions, [13](#)

tile\_by\_windows, [13](#)