Package 'methylSig'

March 10, 2023

Title MethylSig: Differential Methylation Testing for WGBS and RRBS Data

Version 1.11.0

Date 2020-04-22

Description MethylSig 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. MethylSig 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/methylSig/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/methylSig

git_branch master

git_last_commit 5d89bed

git_last_commit_date 2022-11-01

Date/Publication 2023-03-10

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

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

R topics documented:

`bsseq_destranded` *BSseq object read from destranded coverage files*

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 lcous 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 *Performs model fit for general experimental design*

Description

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

Usage

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

Arguments

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

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` *Calculates differential methylation statistics under general experimental design*

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

`bs`	a `BSseq`, the same used used to create `diff_fit`.
`diff_fit`	a `list` object output by `diff_dss_fit()`.
`contrast`	a contrast matrix for hypothesis testing. The number of rows should match the number of columns `design`. Consult `diff_fit$X` to ensure the contrast correponds to the intended test.
`methylation_group_column`	Optionally, a column from `diff_fit$design` 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 methlyation for each group is calculated. If not a `numeric`, use the `methylation_groups` 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` *Calculates differential methylation statistics using a Beta-binomial approach*

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

`bs`	a `BSseq` object.
`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.
`disp_groups`	a named `logical` vector indicating the whether to use `case`, `control`, or both to estimate the dispersion.
`local_window_size`	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.

`local_weight_function`
: a weight kernel function. The default is the tri-weight kernel function defined as `function(u) = (1-u^2)^3`. The domain of any given weight function should be [-1,1], and the range should be [0,1].

`t_approx`
: a `logical` value indicating whether to use squared t approximation for the likelihood ratio statistics. Chi-square approximation (`t_approx = FALSE`) is recommended when the sample size is large. Default is `TRUE`.

`n_cores`
: 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 bsseq::read.bismark 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 `pData(bs)` 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)
```

methylSig *MethylSig: Differential Methylation Testing for WGBS and RRBS Data*

Description

MethylSig 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. MethylSig 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.

methylSig functions

filter_loci_by_coverage() filter_loci_by_snps() tile_by_regions() tile_by_windows() filter_loci_by_group_coverage() diff_binomial() diff_methylsig() diff_methylsig_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/methylSig/issues

---

promoters_gr *GRanges object with collapsed promoters on chr21 and chr22*

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 = 'methylSig')
```

tile_by_regions *Group cytosine / CpG level data into regions based on genomic regions*

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 = 'methylSig')
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 *Group cytosine / CpG level data into regions based on genomic windows*

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