Package 'dmrseq'

November 12, 2026

Type Package

Title Detection and inference of differentially methylated regions from Whole Genome Bisulfite Sequencing

Version 1.33.0

Description This package implements an approach for scanning the genome to detect and perform accurate inference on differentially methylated regions from Whole Genome Bisulfite Sequencing data. The method is based on comparing detected regions to a pooled null distribution, that can be implemented even when as few as two samples per population are available. Region-level statistics are obtained by fitting a generalized least squares (GLS) regression model with a nested autoregressive correlated error structure for the effect of interest on transformed methylation proportions.

License MIT + file LICENSE

Depends R (>= 3.5), bsseq

Imports GenomicRanges, nlme, ggplot2, S4Vectors, RColorBrewer, bumphunter, DelayedMatrixStats (>= 1.1.13), matrixStats, BiocParallel, outliers, methods, locfit, IRanges, grDevices, graphics, stats, utils, annotatr, AnnotationHub, rtracklayer, Seqinfo, splines

Suggests knitr, rmarkdown, BiocStyle, TxDb.Hsapiens.UCSC.hg19.knownGene, org.Hs.eg.db

biocViews ImmunoOncology, DNAMethylation, Epigenetics, MultipleComparison, Software, Sequencing, DifferentialMethylation, WholeGenome, Regression, FunctionalGenomics

LazyData true

VignetteBuilder knitr

RoxygenNote 7.2.3

git_url https://git.bioconductor.org/packages/dmrseq

git_branch devel

git_last_commit dd2a002

git_last_commit_date 2026-04-28

Repository Bioconductor 3.24

Date/Publication 2026-11-12

Author Keegan Korthauer [cre, aut] (ORCID:
<https://orcid.org/0000-0002-4565-1654>),
Rafael Irizarry [aut] (ORCID: <https://orcid.org/0000-0002-3944-4309>),
Yuval Benjamini [aut],
Sutirtha Chakraborty [aut]

Maintainer Keegan Korthauer <keegan@stat.ubc.ca>

Contents

annot.chr21 2
BS.chr21 3
dmrPlotAnnotations 4
dmrs.ex 4
dmrseq 5
getAnnot 7
meanDiff 8
plotDMRs 9
plotEmpiricalDistribution 12
simDMRs 13

Index **15**

`annot.chr21` *annot.chr21: Annotation information for chromosome 21, hg38 genome*

Description

This is the annotation information returned from `getAnnot`, subsetted for chromosome 21 for convenience in running the examples. The annotation is obtained using the `annotatr` package.

Usage

```
data(annot.chr21)
```

Format

a `GRangesList` object with two elements returned by `getAnnot`. The first contains CpG category information in the first element (optional) coding gene sequence information in the second element (optional). At least one of these elements needs to be non-null in order for any annotation to be plotted, but it is not necessary to contain both.

Source

Obtained from running `annoTrack` function and then subsetting the results to only include chromosome 21. A script which executes these steps and constructs the `annot.chr21` object may be found in '`inst/scripts/get_annot.chr21.R`'

Examples

```
data(annot.chr21)
```

BS.chr21 *BS.chr21: Whole-genome bisulfite sequencing for chromosome 21 from Lister et al.*

Description

This dataset represents chromosome 21 from the IMR90 and H1 cell lines sequenced in Lister et al. Only CpG methylation are included. The two samples from each cell line are two different extractions (ie. technical replicates), and are pooled in the analysis in the original paper.

Usage

```
data(BS.chr21)
```

Format

An object of class BSseq.

Details

All coordinates are in hg18.

Source

Obtained from http://neomorph.salk.edu/human_methylome/data.html specifically, the files mc_h1_r1.tar.gz, mc_h1_r2.tar.gz, mc_imr90_r1.tar.gz, mc_imr90_r2.tar.gz A script which downloads these files and constructs the BS.chr21 object may be found in 'inst/scripts/get_BS.chr21.R' - this was based off of and modified from the get_BS.chr22.R script in the bsseq package. The object constructed here contains a different chromosome (22 instead of 21), and two additional samples (h1 and imr90 instead of just imr90) to enable identification of cell type-DMRs for examples.

References

R Lister et al. *Human DNA methylomes at base resolution show widespread epigenomic differences*. Nature (2009) 462, 315-322.

Examples

```
data(BS.chr21)
BS.chr21
```

dmrPlotAnnotations *Add annotations to DMR plots*

Description

Function to add visual representation of CpG categories and/or coding sequences to DMR plot

Usage

```
dmrPlotAnnotations(gr, annoTrack)
```

Arguments

gr	a GRanges object that contains the DMRs to be plotted
annoTrack	a SimpleGRangesList object with two elements. The first contains CpG category information in the first element (optional) coding gene sequence information in the second element (optional). At least one of these elements needs to be non-null in order for any annotation to be plotted, but it is not necessary to contain both.

Details

An internal function that takes an annotation SimpleGRangesList object that contains CpG category information in the first element (optional) and / or coding gene sequence information in the second element (optional). If neither of these are present, then nothing will be plotted.

Value

None

dmrs.ex *dmrs.ex: Example results of DMRs*

Description

Example output from dmrseq function run on the example dataset BS.chr21.

Usage

```
data(dmrs.ex)
```

Format

a data.frame that contains the results of the inference. The data.frame contains one row for each candidate region, and 10 columns, in the following order: 1. chr = chromosome, 2. start = start basepair position of the region, 3. end = end basepair position of the region, 4. indexStart = the index of the region's first CpG, 5. indexEnd = the index of the region's last CpG, 6. L = the number of CpGs contained in the region, 7. area = the sum of the smoothed beta values 8. beta = the coefficient value for the condition difference, 9. stat = the test statistic for the condition difference, 10. pval = the permutation p-value for the significance of the test statistic, and 11. qval = the q-value for the test statistic (adjustment for multiple comparisons to control false discovery rate).

Source

Obtained from running the examples in dmrseq. A script which executes these steps and constructs the dmrs.ex object may be found in 'inst/scripts/get_dmrs.ex.R'

Examples

```
data(dmrs.ex)
```

---

dmrseq *Main function for detecting and evaluating significance of DMRs.*

Description

Performs a two-step approach that (1) detects candidate regions, and (2) scores candidate regions with an exchangeable (across the genome) statistic and evaluates statistical significance using a permuation test on the pooled null distribution of scores.

Usage

```
dmrseq(
  bs,
  testCovariate,
  adjustCovariate = NULL,
  cutoff = 0.1,
  minNumRegion = 5,
  smooth = TRUE,
  bpSpan = 1000,
  minInSpan = 30,
  maxGapSmooth = 2500,
  maxGap = 1000,
  verbose = TRUE,
  maxPerms = 10,
  matchCovariate = NULL,
  BPPARAM = bpparam(),
  stat = "stat",
  block = FALSE,
  blockSize = 5000,
  chrsPerChunk = 1
)
```

Arguments

bs	bsseq object containing the methylation values as well as the phenotype matrix that contains sample level covariates
testCovariate	Character value indicating which variable (column name) in pData(bs) to test for association of methylation levels. Can alternatively specify an integer value indicating which of column of pData(bs) to use. This is used to construct the design matrix for the test statistic calculation. To run using a continuous or categorial covariate with more than two groups, simply pass in the name of a column in 'pData' that contains this covariate. A continuous covariate is assmued if the data type in the 'testCovariate' slot is continuous, with the exception of if there are only two unique values (then a two group comparison is carried out).

adjustCovariate
: an (optional) character value or vector indicating which variables (column names) in pData(bs) will be adjusted for when testing for the association of methylation value with the testCovariate. Can alternatively specify an integer value or vector indicating which of the columns of pData(bs) to adjust for. If not NULL (default), then this is also used to construct the design matrix for the test statistic calculation.

cutoff
: scalar value that represents the absolute value (or a vector of two numbers representing a lower and upper bound) for the cutoff of the single CpG coefficient that is used to discover candidate regions. Default value is 0.10.

minNumRegion
: positive integer that represents the minimum number of CpGs to consider for a candidate region. Default value is 5. Minimum value is 3.

smooth
: logical value that indicates whether or not to smooth the CpG level signal when discovering candidate regions. Defaults to TRUE.

bpSpan
: a positive integer that represents the length in basepairs of the smoothing span window if smooth is TRUE. Default value is 1000.

minInSpan
: positive integer that represents the minimum number of CpGs in a smoothing span window if smooth is TRUE. Default value is 30.

maxGapSmooth
: integer value representing maximum number of basepairs in between neighboring CpGs to be included in the same cluster when performing smoothing (should generally be larger than maxGap)

maxGap
: integer value representing maximum number of basepairs in between neighboring CpGs to be included in the same DMR.

verbose
: logical value that indicates whether progress messages should be printed to stdout. Defaults value is TRUE.

maxPerms
: a positive integer that represents the maximum number of permutations that will be used to generate the global null distribution of test statistics. Default value is 10.

matchCovariate
: An (optional) character value indicating which variable (column name) of pData(bs) will be blocked for when constructing the permutations in order to test for the association of methylation value with the testCovariate, only to be used when testCovariate is a two-group factor and the number of permutations possible is less than 500000. Alternatively, you can specify an integer value indicating which column of pData(bs) to block for. Blocking means that only permutations with balanced composition of testCovariate values will be used (for example if you have samples from different gender and this is not your covariate of interest, it is recommended to use gender as a matching covariate to avoid one of the permutations testing entirely males versus females; this violates the null hypothesis and will decrease power). If not NULL (default), then no blocking is performed.

BPPARAM
: a BiocParallelParam object to specify the parallel backend. The default option is BiocParallel::bpparam() which will automatically creates a cluster appropriate for the operating system.

stat
: a character vector indicating the name of the column of the output to use as the region-level test statistic. Default value is 'stat' which is the region level-statistic designed to be comparable across the genome. It is not recommended to change this argument, but it can be done for experimental purposes. Possible values are: 'L' - the number of loci in the region, 'area' - the sum of the smoothed loci statistics, 'beta' - the effect size of the region, 'stat' - the test statistic for the region, or 'avg' - the average smoothed loci statistic.

block	logical indicating whether to search for large-scale (low resolution) blocks of differential methylation (default is FALSE, which means that local DMRs are desired). If TRUE, the parameters for bpSpan, minInSpan, and maxGapSmooth should be adjusted (increased) accordingly. This setting will also merge candidate regions that (1) are in the same direction and (2) are less than 1kb apart with no covered CpGs separating them. The region-level model used is also slightly modified - instead of a loci-specific intercept for each CpG in theregion, the intercept term is modeled as a natural spline with one interior knot per each 10kb of length (up to 10 interior knots).
blockSize	numeric value indicating the minimum number of basepairs to be considered a block (only used if block=TRUE). Default is 5000 basepairs.
chrsPerChunk	a positive integer value indicating the number of chromosomes per chunk. The default is 1, meaning that the data will be looped through one chromosome at a time. When pairing up multiple chromosomes per chunk, sizes (in terms of numbers of CpGs) will be taken into consideration to balance the sizes of each chunk.

Value

a GRanges object that contains the results of the inference. The object contains one row for each candidate region, sorted by q-value and then chromosome. The standard GRanges chr, start, and end are included, along with at least 7 metadata columns, in the following order: 1. L = the number of CpGs contained in the region, 2. area = the sum of the smoothed beta values 3. beta = the coefficient value for the condition difference (there will be more than one column here if a multi-group comparison was performed), 4. stat = the test statistic for the condition difference, 5. pval = the permutation p-value for the significance of the test statistic, and 6. qval = the q-value for the test statistic (adjustment for multiple comparisons to control false discovery rate). 7. index = an IRanges containing the indices of the region's first CpG to last CpG.

Examples

```
# load example data
data(BS.chr21)

# the covariate of interest is the 'CellType' column of pData(BS.chr21)
testCovariate <- 'CellType'

# run dmrseq on a subset of the chromosome (10K CpGs)
regions <- dmrseq(bs=BS.chr21[240001:250000,],
                  cutoff = 0.05,
                  testCovariate=testCovariate)
```

---

getAnnot *Retrieve annotation information*

Description

Uses the annotatr package to retrieve annotation information (CpG category and gene coding sequences) for the annoTrack argument of plotDMRs. Allows for 5 re-tries if download fails (to allow for a spotty internet connection).

Usage

```
getAnnot(genomeName)
```

Arguments

genomeName	a character object that indicates which organism is under study. Use the function builtin_genomes() to see a character vector of available genome names to choose from (see annotatr documentation for more details).

Details

Note that this package needs to attach the annotatr package, and will return NULL if this cannot be done. You can still use the plotDMRs function without this optional annotation step, just by leaving the annoTrack argument as NULL.

Value

a SimpleGRangesList object with two elements returned by getAnnot. The first contains CpG category information in the first element (optional) coding gene sequence information in the second element (optional). At least one of these elements needs to be non-null in order for any annotation to be plotted, but it is not necessary to contain both.

Examples

```
# get annotation information for hg19
annoTrack <- getAnnot('hg19')
```

---

meanDiff *Function to compute raw mean methylation differences*

Description

This function calculates raw mean methylation differences for the covariate of interest over a set of DMRs (or regions of interest), assuming a simple two-group comparison.

Usage

```
meanDiff(bs, dmrs, testCovariate)
```

Arguments

bs	a BSseq object
dmrs	a data.frame with one row per DMR. This can be in the format of dmrseq output, but at least should contain the indexStart and indexEnd values of the regions of interest.
testCovariate	a character indicating the covariate of interest in the pData slot of bs.

Value

numeric vector of raw mean methylation differences.

Examples

```
data(BS.chr21)
data(dmrs.ex)
rawDiff <- meanDiff(BS.chr21, dmrs=dmrs.ex, testCovariate="CellType")
```

`plotDMRs` *Plot Differentially Methylated Regions*

Description

Generates trace plots of methylation proportions by genomic position.

Usage

```
plotDMRs(
  BSseq,
  regions = NULL,
  testCovariate = NULL,
  extend = (end(regions) - start(regions) + 1)/2,
  main = "",
  addRegions = regions,
  annoTrack = NULL,
  col = NULL,
  lty = NULL,
  lwd = NULL,
  label = NULL,
  mainWithWidth = TRUE,
  regionCol = .alpha("#C77CFF", 0.2),
  addTicks = TRUE,
  addPoints = TRUE,
  pointsMinCov = 1,
  highlightMain = FALSE,
  qval = TRUE,
  stat = TRUE,
  verbose = TRUE,
  includeYlab = TRUE,
  compareTrack = NULL,
  labelCols = NULL,
  horizLegend = FALSE,
  addLines = TRUE,
  linesMinCov = 1
)
```

Arguments

`BSseq`	An object of class BSseq.
`regions`	A data.frame containing the DMRs (output from the main `dmrseq`) function.
`testCovariate`	integer value or vector indicating which of columns of `pData(bs)` contains the covariate of interest. This is used to construct the sample labels and colors (unless this is over-ridden by specifying `label`).

`extend` Describes how much the plotting region should be extended in either direction. The total width of the plot is equal to the width of the region plus twice extend.

`main` The plot title. The default is to construct a title with information about which genomic region is being plotted.

`addRegions` A set of additional regions to be highlighted on the plots. Same format as the `regions` argument.

`annoTrack` a `GRangesList` object with two elements returned by `getAnnot`. The first contains CpG category information in the first element (optional) coding gene sequence information in the second element (optional). At least one of these elements needs to be non-null in order for any annotation to be plotted, but it is not necessary to contain both.

`col` The color of the methylation estimates. It is recommended to leave this value as default (NULL), and specify a value of `testCovariate` to indicate which column of `pData(bs)` to use as a factor for coloring the points and lines of the plot. Alternatively, you can specify particular colors by passing this information through the `pData` slot of the object `BSseq` (a `data.frame` that houses metadata). To do so, place the color value for each sample in a column titled `col`, and leave this argument as its default value of NULL. Alternatively, you may specify a vector of color values (one for each sample), but you *must* make sure that this vector is in the same order as the samples are in the BSseq object. If NULL and no `col` column is found in `pData`, then estimates are plotted in black for all samples.

`lty` The line type of the methylation estimates. It is recommended to pass this information through the `pData` slot of the object `BSseq` (a `data.frame` that houses metadata). To do so, place the line type value for each sample in a column titled `lty`, and leave this argument as its default value of NULL. Alternatively, you may specify a vector of line type values (one for each sample), but you *must* make sure that this vector is in the same order as the samples are in the BSseq object. If NULL and no `lty` column is found in `pData`, then estimates are plotted with `lty=1` for all samples.

`lwd` The line width of the methylation estimates. It is recommended to pass this information through the `pData` slot of the object `BSseq` (a `data.frame` that houses metadata). To do so, place the line width value for each sample in a column titled `lwd`, and leave this argument as its default value of NULL. Alternatively, you may specify a vector of line width values (one for each sample), but you *must* make sure that this vector is in the same order as the samples are in the BSseq object. If NULL and no `lwd` column is found in `pData`, then estimates are plotted with `lwd=1` for all samples.

`label` The condition/population labels for the plot legend. If NULL (default) this is taken from the `testCovariate` column of `pData`. Alternatively, you can pass in labels by adding this information through the `pData` slot of the object `BSseq` (a `data.frame` that houses metadata). To do so, place the labels for each sample in a column titled `label`, and leave this argument as its default value of NULL. You may instead specify an arbitrary vector of labels (one for each sample), but be aware that you *must* make sure that this vector is in the same order as the samples are in the BSseq object. If NULL, and `testCovariate` is also NULL and no `label` column is found in `pData`, then no legend is created.

`mainWithWidth` logical value indicating whether the default title should include information about width of the plot region.

`regionCol` The color used for highlighting the region.

addTicks — logical value indicating whether tick marks showing the location of methylation loci should be added. Default is TRUE.

addPoints — logical value indicating whether the individual methylation estimates be plotted as points.

pointsMinCov — The minimum coverage a methylation loci need in order for the raw methylation estimates to be plotted. Useful for filtering out low coverage loci. Only used if addPoints = TRUE. Default value is 1 (no filtering).

highlightMain — logical value indicating whether the plot region should be highlighted.

qval — logical value indicating whether the region FDR estimate (q-value) should be displayed in the plot title. The value is extracted from the regions argument.

stat — logical value indicating whether the region statistic should be displayed in the plot title. The value is extracted from the regions argument.

verbose — logical value indicating whether progress messages should be printed to the screen.

includeYlab — a logical indicating whether to include the Y axis label 'Methylation' (useful to turn off if combining multiple region figures and you do not want to include redundant y axis label information)

compareTrack — a named GRangesList object that contains up to four custom tracks (GRanges objects) which will be plotted below the region. Only one of 'compareTrack' or 'annoTrack' can be specified since there is only for plotting either the built in GpG category and exon tracks, *or* a custom set of tracks.

labelCols — a character vector with names of the mcols slot of the GRanges items in 'compareTrack'. Only used if plotting custom tracks using the 'compareTrack' argument. If specified, the (first) value in that column is printed along with a label that includes the name of the list item. If NULL (default), just the name of the track is printed.

horizLegend — logical indicating whether the legend should be horizontal instead of vertical (default FALSE). This is useful if you need to plot many labels and want to preserve whitespace.

addLines — logical indicating whether to plot smooth lines between points. Default is true. Can be useful to turn this off for very small regions.

linesMinCov — The minimum coverage a methylation loci need in order to be used for plotting of smoothed lines. Useful for filtering out low coverage loci. Only used if addLines = TRUE. Default value is 1 (no filtering).

Details

Creates aesthetially pleasing DMR plots. By default will plot individual points with size proportional to coverage, along with a smoothed line for each sample. Elements will be colored by biological condition (label). Also has functionality to add annotations below the main plot (CpG category, genes) if annoTrack is specified.

Value

None (generates a plot)

Examples

```
# load the example data
data(BS.chr21)

# load example results (computed with dmrseq function)
data(dmrs.ex)

# get annotation information (using getAnnot function)
# here we'll load the example annotation from chr21
data(annot.chr21)

# plot the 1st DMR
plotDMRs(BS.chr21, regions=dmrs.ex[1,], testCovariate=1,
   annoTrack=annot.chr21)
```

`plotEmpiricalDistribution` *Plot the empirical distribution of the methylation beta vals or coverage*

Description

Uses ggplot2 to plot smoothed density histograms of methylation proportions (beta values), or coverage. Methylation proportion densities are weighted by coverage. The number of curves plotted will be equal to the number of different values of `testCovariate`, unless `bySample` is TRUE. This can take quite some time to execute for a large object, so it is recommended to first take a random sample of loci (say one million) before plotting.

Usage

```
plotEmpiricalDistribution(
  bs,
  testCovariate = NULL,
  bySample = FALSE,
  type = "M",
  adj = 2.5
)
```

Arguments

`bs`	a BSseq object
`testCovariate`	character specifying the column name of the `pData` slot of the BSseq object to include in the plot legend.
`bySample`	logical whether to plot a separate line for each sample, even if the grouping `testCovariate` is specified. Default value is FALSE (so samples with the same value of `testCovariate` will be collapsed into the same line). If `testCovariate` is not specified, this parameter does not have an effect and samples are automatically plotted separately.
`type`	a character indicating which type of density to plot - the methylation (beta) values ("M") or the coverage ("Cov"). Default is "M".
`adj`	a numeric value for the `adjust` parameter to pass to the `geom_line` function. Specifies how smooth the make the function.

Value

a ggplot object

Examples

```
data(BS.chr21)

# plot beta values by sample group
plotEmpiricalDistribution(BS.chr21, testCovariate="CellType")
```

`simDMRs` *Simulate Differentially Methylated Regions*

Description

Add simulated DMRs to observed control data. Control data will be split into two (artificial) populations.

Usage

```
simDMRs(bs, num.dmrs = 3000, delta.max0 = 0.3)
```

Arguments

`bs`	a BSseq object containing only control samples (from the same population) for which simulated DMRs will be added after dividing the population into two artificial groups.
`num.dmrs`	an integer specifying how many DMRs to add.
`delta.max0`	a proportion value indicating the mode value for the difference in proportion of methylated CpGs in the simulated DMRs (the actual value will be drawn from a scaled Beta distribution centered at this value). Default value is 0.3.

Value

A named list object with 5 elements: (1) `gr.dmrs` is a `GRanges` object with `num.dmrs` ranges that represent the random DMRs added. (2) `dmr.mncov` is a numeric vector that contains the mean coverage in each simulated DMR. (3) `dmr.L` is a numeric vector that contains the number of CpGs in each simulated DMR. (4) `bs` is the BSseq object that contains the simulated DMRs. (5) `deltas` is a numeric vector that contains the effect size used for each DMR.

Examples

```
# Add simulated DMRs to a BSseq dataset
# This is just for illustrative purposes - ideally you would
# add DMRs to a set of samples from the same condition (in our
# example data, we have data from two different cell types)
# In this case, we shuffle the samples by cell type to create
# a null comparison.

data(BS.chr21)
```

```
BS.chr21.sim <- simDMRs(bs=BS.chr21[1:10000,c(1,3,2,4)],
                        num.dmrs=50)

# show the simulated DMRs GRanges object
show(BS.chr21.sim$gr.dmrs)

# show the updated BSseq object that includes the simulated DMRs
show(BS.chr21.sim$bs)

# examine effect sizes of the DMRs
head(BS.chr21.sim$delta)
```

Index

∗ **datasets**
annot.chr21, 2
BS.chr21, 3
dmrs.ex, 4
∗ **inference**
dmrseq, 5

annot.chr21, 2

BS.chr21, 3

dmrPlotAnnotations, 4
dmrs.ex, 4
dmrseq, *5*, 5

getAnnot, *2*, 7, *8*, *10*

meanDiff, 8

plotDMRs, *7, 8*, 9
plotEmpiricalDistribution, 12

simDMRs, 13