Package 'CMEnt'

November 13, 2026

Type Package

Title Characterization of Methylation using positional ENTanglement

Version 0.99.15

Description CMEnt implements a correlation-based method for identifying Differentially Methylated Regions (DMRs) from genomic seeds, commonly Differentially Methylated Positions (DMPs). The package expands regions around significant seeds considering both statistical significance and biological relevance of methylation changes. It supports array-based (450K, EPIC, EPICv2) and NGS methylation data through tabix-indexed files, provides DMR interaction analysis via motif similarity, and includes comprehensive visualization tools including circos plots and beta value heatmaps.

URL https://cmg-ua.github.io/CMEnt/, https://github.com/CMG-UA/CMEnt

BugReports https://github.com/CMG-UA/CMEnt/issues

License GPL (>= 2)

SystemRequirements tabix, bgzip, GNU sort

Encoding UTF-8

Roxygen list(markdown = TRUE)

VignetteBuilder knitr

biocViews DNAMethylation, DifferentialMethylation, Epigenetics, StatisticalMethod, WorkflowStep, Annotation, MethylationArray, MotifAnnotation

Depends R (>= 4.6.0)

LazyData false

Imports AnnotationDbi, bedr, BiocGenerics, BiocFileCache, BiocParallel, Biostrings, BSgenome, bslib (>= 0.5.0), bsseq, callr, circlize, cli, colorspace, ComplexHeatmap, data.table, DelayedArray, DelayedDataFrame, DT, e1071, filelock, FNN, futile.logger, GenomeInfoDb, GenomicFeatures, GenomicRanges, ggplot2, ggseqlogo, graphics, grDevices, gridBase, gridExtra, HDF5Array, igraph, inline, IRanges, jsonlite, limma, matrixStats, methods, plotly, R.utils, R6, rhdf5, reshape2, rtracklayer, S4Vectors, shiny (>= 1.7.0), shinycssloaders, showtext, stats, strex, stringr, SummarizedExperiment, TFBSTools, tools, utils, withr

Suggests bsseqData, DMRsegaldata, ExperimentHub, ggnewscale, IlluminaHumanMethylation27kanno.ilmn12.hg19, IlluminaHumanMethylation450kanno.ilmn12.hg19, IlluminaHumanMethylationEPICanno.ilm10b4.hg19, IlluminaHumanMethylationEPICv2anno.20a1.hg38, BiocManager, devtools, DSS, knitr, pkgdown, rmarkdown, BiocStyle, magick, minfi, testthat (>= 3.0.0), mockery, DMRcate, doParallel, bumphunter, tidyr, scales, UpSetR, TxDb.Hsapiens.UCSC.hg19.knownGene, TxDb.Hsapiens.UCSC.hg38.knownGene, BSgenome.Hsapiens.UCSC.hg19, BSgenome.Hsapiens.UCSC.hg38, BSgenome.Hsapiens.UCSC.hs1, BSgenome.Mmusculus.UCSC.mm10, BSgenome.Mmusculus.UCSC.mm39, org.Hs.eg.db, org.Mm.eg.db, JASPAR2024, optparse, pkgload

Config/testthat/edition 3

Collate 'annotate_dmrs_with_genes.R' 'augment_bsseq.R' 'beta_handler.R' 'CMEnt-package.R' 'combine_pvalues.R' 'convert_beta_to_tabix.R' 'dmrs_builder.R' 'dmrs_interaction.R' 'find_dmps_array.R' 'find_dmps_bsseq.R' 'get_dmr_sequences.R' 'load_example_input_data.R' 'plot_dmrs.R' 'plot_dmrs_circos.R' 'read_custom_methylation_bed_data.R' 'reduce_bsseq_around_seeds.R' 'shiny_app.R' 'shiny_modules.R' 'score_dmrs.R' 'simulate_dmrs.R' 'sparsify_seeds.R' 'sort_beta_file_by_coordinates.R' 'utils.R' 'zzz.R'

NeedsCompilation no

Config/roxygen2/version 8.0.0

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

git_branch devel

git_last_commit 6309a64

git_last_commit_date 2026-10-19

Repository Bioconductor 3.24

Date/Publication 2026-11-13

Author Vasileios Lemonidis [aut, cre, cph] (ORCID: <https://orcid.org/0000-0002-6446-3536>), Center for Oncological Research, University of Antwerp [cph, fnd], Stichting Tegen Kanker [fnd]

Maintainer Vasileios Lemonidis <vasileios.lemonidis@uantwerpen.be>

Contents

CMEnt-package 3
annotateDMRsWithGenes 5
augmentBSSeq 6
BetaHandler 8
buildDMRs 11
combinePvalues 15
computeDMRsInteraction 17
convertBetaToTabix 18
extractDMRMotifs 20

findDMPsArray 21
findDMPsBSSeq 23
getBetaHandler 24
getDMRSequences 25
getSortedGenomicLocs 27
launchCMEntViewer 28
loadExampleInputData 29
orderByLoc 30
plotAutoDMRsCircos 31
plotDMR 33
plotDMRBlockFormation 34
plotDMRs 36
plotDMRsCircos 37
plotDMRsManhattan 40
readCustomMethylationBedData 41
reduceBSseqAroundSeeds 43
scoreDMRs 44
simulateDMRs 46
sortBetaFileByCoordinates 49
sparsifySeeds 51

Index **52**

`CMEnt-package` *CMEnt: Characterization of Methylation using positional ENTanglement*

Description

CMEnt is a comprehensive tool for identifying Differentially Methylated Regions (DMRs) from genomic seeds, commonly Differentially Methylated Positions (DMPs). The package uses a correlation-based approach to expand regions around significant seeds, considering both statistical significance and biological relevance of methylation changes.

Main Functions

- `buildDMRs`: Identify DMRs from genomic seeds
- `extractDMRMotifs`: Extract sequence motifs from DMRs
- `computeDMRsInteraction`: Compute motif-based DMR interactions
- `plotDMR`: Visualize individual DMRs with structure, beta values, and motifs
- `plotDMRs`: Create multi-panel DMR plots
- `plotDMRsCircos`: Generate circos plots with DMR interactions
- `plotDMRsManhattan`: Generate genome-wide Manhattan-style plots for DMR scores

Input Data

The package accepts multiple types of methylation data input:

- Beta value files: Tab-separated files containing methylation beta values
- Tabix files: Indexed bed.gz files for efficient access to large datasets
- BED files: Custom methylation BED files (automatically converted to tabix)
- Beta matrices: In-memory beta value matrices

Platform Support

The package supports methylation data from multiple platforms:

- Human arrays: 450K, EPIC (850k), EPICv2, 27K on hg19, hg38, or hs1
- Mouse arrays: mm10, mm39 genomes
- NGS data: Via tabix-indexed files
- Custom platforms: Via BED file input

Workflow

1. Start with genomic seeds (e.g., DMPs) from any differential analysis
2. Provide methylation data (multiple formats supported)
3. Define parameters for region expansion and connectivity
4. Get DMRs as a GRanges object with comprehensive statistics
5. Optionally extract motifs and compute DMR interactions
6. Visualize with structure plots, heatmaps, and circos plots

Author(s)

Maintainer: Vasileios Lemonidis <vasileios.lemonidis@uantwerpen.be> (ORCID) [copyright holder]

Authors:

- Vasileios Lemonidis <vasileios.lemonidis@uantwerpen.be> (ORCID) [copyright holder]

Other contributors:

- Center for Oncological Research, University of Antwerp [copyright holder, funder]
- Stichting Tegen Kanker [funder]

See Also

Useful links:

- https://cmg-ua.github.io/CMEnt/
- https://github.com/CMG-UA/CMEnt
- Report bugs at https://github.com/CMG-UA/CMEnt/issues

`annotateDMRsWithGenes` *Annotate DMRs with Gene Information*

Description

Annotates DMRs with overlapping gene promoters and gene bodies using TxDb annotations. For each DMR, identifies genes whose promoters or gene bodies overlap with the DMR coordinates.

Usage

```
annotateDMRsWithGenes(
  dmrs,
  genome = NULL,
  promoter_upstream = 2000,
  promoter_downstream = 200,
  njobs = getOption("CMEnt.njobs", .defaultNJobs()),
  site_locs = NULL,
  site_delta_beta = NULL,
  aggfun = stats::median
)
```

Arguments

`dmrs`	Dataframe or GRanges object containing DMR coordinates
`genome`	Character. Genome version to use for gene annotation. Required for data.frame DMRs; inferred from GRanges seqinfo when available.
`promoter_upstream`	Integer. Number of base pairs upstream of TSS to define promoter region (default: 2000)
`promoter_downstream`	Integer. Number of base pairs downstream of TSS to define promoter region (default: 200)
`njobs`	Integer. Number of parallel jobs used to annotate promoter and gene-body overlaps (default: `getOption("CMEnt.njobs")`)
`site_locs`	Optional data frame or GRanges with site coordinates used to compute feature-specific delta beta values.
`site_delta_beta`	Optional named numeric vector of per-site delta beta values.
`aggfun`	Function used to aggregate per-site delta beta values.

Details

The function uses genome-appropriate TxDb packages. For `hs1`, CMEnt uses hg38 gene models and lifts them to `hs1` before computing overlaps. Gene symbols are retrieved from the appropriate org.*.eg.db package. Multiple overlapping genes are concatenated with commas.

Value

The input Dataframe/GRanges object with additional metadata columns:

- in_promoter_of: Character vector of gene symbols with promoters overlapping the DMR (comma-separated)
- in_gene_body_of: Character vector of gene symbols with gene bodies overlapping the DMR (comma-separated)
- delta_beta_promoter: Aggregated delta beta of DMR sites overlapping promoters, or NA
- delta_beta_gene_body: Aggregated delta beta of DMR sites overlapping gene bodies, or NA

Examples

```
# Annotate DMRs with gene information
dmrs <- data.frame(
    chr = c("chr1", "chr2"),
    start = c(1000000, 2000000),
    end = c(1001000, 2001000)
)
dmrs_annotated <- annotateDMRsWithGenes(dmrs, genome = "hg38")

# Use custom promoter definition
dmrs_annotated <- annotateDMRsWithGenes(
    dmrs,
    genome = "hg38",
    promoter_upstream = 5000,
    promoter_downstream = 1000,
    njobs = 2
)
```

---

augmentBSSeq *Augment BSseq Object*

Description

Generate synthetic samples while preserving both per-site coverage and methylation marginals and the local correlation structure of neighboring sites. Coverage and methylation are first fit with per-site Poisson/Beta marginals, then synthetic samples are drawn through chromosome-local Gaussian copula fields whose dependence is estimated from adjacent sites in the observed data. For reproducible synthetic samples, call `set.seed()` before `augmentBSSeq()`.

Usage

```
augmentBSSeq(
  bs,
  n_new_samples,
  min_samples = 2,
  calibrate_correlation = TRUE,
  calibration_iterations = 8,
  calibration_samples = NULL
)
```

Arguments

bs	A BSseq object
n_new_samples	Number of new synthetic samples to generate
min_samples	Minimum number of samples with coverage required per site
calibrate_correlation	Logical. If TRUE, iteratively adjusts the latent Gaussian length scales so adjacent-site correlations are matched after transforming back through the observed coverage and methylation sampling layers.
calibration_iterations	Maximum number of bisection iterations used for each correlation calibration.
calibration_samples	Number of synthetic samples used internally for correlation calibration. If NULL, a capped conservative default is chosen from the input and requested output sample sizes.

Details

The input is first sorted, duplicate loci are collapsed by summing methylated and coverage counts, and sites are retained only when at least `min_samples` samples have positive coverage. Synthetic coverage and methylation are then modeled separately, but with genomic dependence restored through latent Gaussian fields.

For coverage, each site `i` receives a Poisson mean `lambda_i = mean(C_ij | C_ij > 0)`, with a fallback of 1. A chromosome-local latent Gaussian field `Z^C` is converted to uniforms by `Phi(Z^C_i)` and then to coverage counts by `qpois(Phi(Z^C_i), lambda_i)`, with a lower bound of 1 for synthetic samples.

For methylation, covered observations are smoothed as `p_ij = (M_ij + 0.5) / (C_ij + 1)`. The site mean is estimated with the same Jeffreys-style pseudocounts, and the across-sample variance of `p_ij` is converted to a Beta concentration `kappa_i = p_i * (1 - p_i) / var(p_ij) - 1`. Site concentrations are shrunk toward an empirical prior, then bounded to avoid extreme under- or over-dispersion. This gives `alpha_i = p_i * kappa_i` and `beta_i = (1 - p_i) * kappa_i`, with small positive lower bounds.

Local correlation is represented by an exponential Gaussian copula. For adjacent sites separated by gap `g`, the latent correlation is `phi = exp(-g / ell)`, so within each chromosome `Z_i = phi_i * Z_{i-1} + sqrt(1 - ph` Separate length scales `ell` are estimated for coverage from adjacent-site correlations of `log1p(C)` and for methylation from adjacent-site correlations of `qlogis(p_ij)`.

If `calibrate_correlation = TRUE`, the initial length scales are refined by bisection against the observed median adjacent-site correlation after passing simulated latent pairs through the same Poisson, Beta, and Binomial sampling layers used for final augmentation. This compensates for correlation attenuation caused by the marginal sampling steps.

Final synthetic samples are generated as `C_i ~ Poisson(lambda_i)`, `theta_i ~ Beta(alpha_i, beta_i)`, and `M_i ~ Binomial(C_i, theta_i)`, with both `C_i` and `theta_i` driven by their calibrated chromosome-local latent fields. The returned object combines the retained original sites and samples with the synthetic samples.

Value

A BSseq object with original and synthetic samples

Examples

```
if (requireNamespace("bsseqData", quietly = TRUE)) {
    data(BS.cancer.ex, package = "bsseqData")
    BS.cancer.ex <- BS.cancer.ex[seq_len(100), ]
    set.seed(123)
    augmented_bs <- augmentBSSeq(BS.cancer.ex, n_new_samples = 2)
}
```

`BetaHandler` *Beta Handler Class*

Description

An R6 class to handle methylation beta value files efficiently, with support for in-memory loading, tabix indexing, BSseq objects, and various file formats.

Public fields

`beta` Path to beta values file, or a tabix file, or in-memory beta matrix, or BSseq object

`genome` Reference genome

`array` Array platform, ignore for mouse genomes or when sorted_locs provided

`beta_row_names_file` Path to row names file

`sorted_locs` Sorted genomic locations

`output_prefix` Prefix used for saving derived beta artifacts

`njobs` Number of parallel jobs

`beta_chunk_size` Chunk size for subsetting beta values

Methods

Public methods:

- `BetaHandler$new()`
- `BetaHandler$load()`
- `BetaHandler$getGenomicLocs()`
- `BetaHandler$getBetaRowNames()`
- `BetaHandler$getBetaColNames()`
- `BetaHandler$validate()`
- `BetaHandler$getBetaLocs()`
- `BetaHandler$getBetaChunkSize()`
- `BetaHandler$isArrayBased()`
- `BetaHandler$subset()`
- `BetaHandler$getBeta()`
- `BetaHandler$clone()`

`BetaHandler$new()`: Create a new BetaHandler object

Usage:

```
BetaHandler$new(
  beta = NULL,
  array = NULL,
  genome = NULL,
  beta_row_names_file = NULL,
  sorted_locs = NULL,
  chrom_col = "#chrom",
  start_col = "start",
  output_prefix = NULL,
  njobs = 1
)
```

Arguments:

`beta` Path to beta values file, or a tabix, or a beta matrix, or a BSseq object

`array` Array platform type. Ignored if sorted_locs, a tabix file, or a BSseq object have been provided.

`genome` Reference genome version, eg. hg38 or hs1. Only human and mouse genomes are supported. Ignored if sorted_locs, a tabix file, or a BSseq object have been provided.

`beta_row_names_file` Path to row names file. If NULL, row names will be read from input `beta`.

`sorted_locs` Sorted genomic locations data frame. If given, the input data will be assumed already sorted. If NULL, will be inferred from the input.

`chrom_col` Chromosome column name in tabix file

`start_col` Start position column name in tabix file

`output_prefix` Prefix used for saving derived beta artifacts

`njobs` Number of parallel jobs

Returns: A new BetaHandler object

`BetaHandler$load()`**:** Load beta file data into memory or prepare for file-based access

Usage:

```
BetaHandler$load()
```

Returns: Self (invisibly)

`BetaHandler$getGenomicLocs()`**:** Get **all** provided sorted genomic locations

Usage:

```
BetaHandler$getGenomicLocs()
```

Returns: data.frame or matrix of the genomic locations

`BetaHandler$getBetaRowNames()`**:** Get row names (site IDs) from the beta file

Usage:

```
BetaHandler$getBetaRowNames()
```

Returns: Character vector of row names

`BetaHandler$getBetaColNames()`**:** Get column names (sample IDs) from the beta file

Usage:

```
BetaHandler$getBetaColNames()
```

Returns: Character vector of column names

`BetaHandler$validate()`**:** Validate that the beta file is properly sorted and formatted

Usage:

```
BetaHandler$validate()
```

Returns: Self (invisibly)

`BetaHandler$getBetaLocs()`: Get genomic locations for beta values

Usage:

```
BetaHandler$getBetaLocs()
```

Returns: Data frame of genomic locations

`BetaHandler$getBetaChunkSize()`: Get the chunk size used for beta subsetting

Usage:

```
BetaHandler$getBetaChunkSize()
```

Returns: Integer chunk size

`BetaHandler$isArrayBased()`: Check if the beta data is array-based (i.e. does not have row names in 'chr:pos' format)

Usage:

```
BetaHandler$isArrayBased()
```

Returns: Logical indicating if the beta data is array-based

`BetaHandler$subset()`: Build a compact BetaHandler view for a subset of rows/columns

Usage:

```
BetaHandler$subset(row_names = NULL, col_names = NULL, allow_missing = FALSE)
```

Arguments:

`row_names` Character vector of site IDs (or numeric row indices) to keep; NULL keeps all rows

`col_names` Character vector of sample IDs to keep; NULL keeps all columns

`allow_missing` Logical; if TRUE, silently drops missing row names

Returns: A BetaHandler object scoped to the requested subset

`BetaHandler$getBeta()`: Extract beta values for specific site sites and samples

Usage:

```
BetaHandler$getBeta(
  row_names = NULL,
  col_names = NULL,
  allow_missing = FALSE,
  chr = NULL
)
```

Arguments:

`row_names` Character vector of site IDs to extract. If numeric, treated as row indices.

`col_names` Character vector of sample IDs to extract (default: NULL for all)

`allow_missing` Logical. If TRUE, missing site sites will be ignored instead of throwing an error (default: FALSE)

`chr` Character vector of chromosome names to extract, cannot be used along with row_names (default: NULL for all)

Returns: Matrix-like object of beta values

`BetaHandler$clone()`: The objects of this class are cloneable with this method.

Usage:

```
BetaHandler$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

`buildDMRs` *Build Differentially Methylated Regions (DMRs) from Differentially Methylated Positions (seeds)*

Description

This function assembles DMRs from a given set of seeds and a beta value file. It operates in three main stages:

1. **Seed Connectivity**: It builds a connectivity array based on the correlation of beta values between seeds and their proximal sites, connecting seeds into preliminary DMRs based on significant correlations.
2. **DMR Expansion**: It expands the preliminary DMRs by including nearby sites that show significant correlation with the seeds, allowing for a specified delta-beta threshold to connect sites that may not meet the correlation p-value cutoff but have a strong effect size.
3. **DMR Merging and Filtering**: It merges overlapping extended DMRs into final DMRs and applies filtering based on the number of seeds and sites, as well as optional adjustments for array-based analyses.

Usage

```
buildDMRs(
  beta,
  seeds,
  pheno,
  seeds_id_col = NULL,
  sample_group_col = NULL,
  casecontrol_col = NULL,
  covariates = NULL,
  ext_site_delta_beta = 0.2,
  array = NULL,
  genome = NULL,
  max_pval = 0.05,
  entanglement = c("weak", "strong"),
  testing_mode = c("auto", "parametric", "empirical"),
  empirical_strategy = c("auto", "montecarlo", "permutations"),
  ntries = 200L,
  mid_p = FALSE,
  max_lookup_dist = 10000,
  expansion_window = "auto",
  max_bridge_seeds_gaps = 1L,
  max_bridge_extension_gaps = 1L,
  min_seeds = 2,
  min_adj_seeds = NULL,
  min_sites = 3,
```

```
  aggfun = c("median", "mean"),
  ignored_sample_groups = NULL,
  output_prefix = NULL,
  njobs = getOption("CMEnt.njobs", .defaultNJobs()),
  beta_row_names_file = NULL,
  annotate_with_genes = TRUE,
  score_dmrs = TRUE,
  extract_motifs = TRUE,
  bed_provided = FALSE,
  bed_chrom_col = "chrom",
  bed_start_col = "start",
  verbose = getOption("CMEnt.verbose", 1),
  .load_debug = FALSE
)
```

Arguments

`beta`	Character. Path to the beta value file, or a tabix file, or a beta matrix, or a BetaHandler object, or a bed file. If a bed file is provided, it must at least contain bed_chrom_col and bed_chrom_start, followed by samples names in the given pheno, with corresponging beta values, and it will be converted to a tabix-indexed beta file internall, with the locations separately saved and queried as a DelayedDataFrame. object.
`seeds`	Character. Path to the seeds (seeds, etc.) TSV file or the seeds dataframe, in a format like the one produced by dmpFinder. If a `pval`, `P.Value`, `p.value`, or `p_value` column is present, DMR-level `pval` is computed from final supporting seed p-values using Stouffer's method and `qval` is FDR-corrected globally.
`pheno`	Character. Path to the phenotype TSV file or the phenotype dataframe, containing sample information including group labels and optionally covariates.
`seeds_id_col`	Character. Column name or index for Seed identifiers in the seeds TSV file. Default is NULL, which corresponds to the rows names if existing, or the first column if not.
`sample_group_col`	Character. Required column name for sample group information in the phenotype data.
`casecontrol_col`	Boolean Column in pheno for case (TRUE/1) / control (FALSE/0) status . If NULL, controls will be assumed to be the first level of sample_group_col. Default is NULL.
`covariates`	Character vector of column names in pheno to adjust for (e.g. "age", "sex"). When provided, correlations are computed on residuals after regressing M-values on these covariates within each group. Parametric tests use `n - rank(X) - 1` degrees of freedom, where `X` is the within-group covariate design matrix including its intercept.
`ext_site_delta_beta`	Numeric. Minimum absolute delta beta value that will force proximal sites to be treated as connected during Stage 2 expansion, regardless of their correlation p-value. Set to `NA`, `NULL`, or `Inf` to disable this shortcut. A value of `0` means any proximal site with a non-missing case-control delta beta can be force-connected. Default is 0.2.

`array` Character. Type of array used (e.g., "450K", "EPIC", "EPICv2", "27K", "Mouse"). Required when beta row names are array probe IDs. Ignored if the beta file is provided as a beta values BED file.

`genome` Optional character genome version. If `NULL`, inferred from `beta` when possible, otherwise from `array`; BED inputs default to `"hg38"`.

`max_pval` Numeric. Maximum p-value to assume seeds correlation is significant. Default is 0.05.

`entanglement` Character. "weak" (default) requires at least one group to show significant correlation; "strong" requires all groups to show significant correlation for connectivity. Constant sites do not provide a passing group correlation in weak mode.

`testing_mode` Character. "auto" (default) selects between t-based correlation p-values and empirical p-values per sample group using data diagnostics. You can also force "parametric" for t-based correlation p-values or "empirical" for permutation-based p-values.

`empirical_strategy`
Character. When testing_mode = "empirical": "auto" (default) uses Monte Carlo for groups with <6 samples and permutations for groups with >=6 samples; "montecarlo" always uses Monte Carlo; "permutations" always uses permutations.

`ntries` Integer. Number of permutations when testing_mode = "empirical". Default is 0 (disabled).

`mid_p` Logical. Whether to use mid-p values for empirical correlation tests. Default is FALSE.

`max_lookup_dist`
Numeric. Maximum distance to look up for adjacent seeds belonging to the same DMR during Stage 1. Default is 10000 (10 kb).

`expansion_window`
Numeric. Stage 2 connectivity is computed only in windows centered on seed-derived Stage 1 DMR neighborhoods, with this total window width in bp. This value sets a maximum effective size of a DMR after stage 2. Set <=0 for genome-wide connectivity. Default is -1 for microarrays and 10000 (10 kb) for NGS datasets.

`max_bridge_seeds_gaps`
Integer. Maximum number of consecutive failed seed-to-seed edges to bridge during Stage 1 when both flanking edges are connected and failures are p-value driven. Set to 0 to disable. Default is 1.

`max_bridge_extension_gaps`
Integer. Maximum gap size to consider during Stage 2 extension. Default is 1 (i.e., at most 1 consecutive failing site to bridge).

`min_seeds` Numeric. Minimum number of connected seeds in a DMR. Minimum is 1. Default is 2.

`min_adj_seeds` Numeric. Minimum number of seeds, adjusted by array site density, in a DMR after extension. It serves as a less stringent cutoff for arrays with variable site density, allowing regions in sparse areas to be retained if they have enough seeds relative to the local site density. Default is NULL (disabled).

`min_sites` Numeric. Minimum number of sites in a DMR after extension, including the seeds. Minimum is 2. Default is 3.

`aggfun` Function or character. Aggregation function to use when calculating delta beta values and p-values of DMRs. Can be "median", "mean", or a function (e.g., median, mean). Default is "median".

`ignored_sample_groups`
: Character vector. Sample groups to ignore during connection and expansion, separated by commas. Can also be "case" or "control". Default is NULL.

`output_prefix`
: Character. Identifier for the output files. If not provided, no output will be saved. Default is NULL.

`njobs`
: Numeric. Number of parallel jobs to use. Default is the number of available cores.

`beta_row_names_file`
: Character. Path to a file containing row names for the beta values. If not provided, row names will be read from the beta file. Default is NULL.

`annotate_with_genes`
: Logical. Whether to annotate DMRs with overlapping genes. Default is TRUE.

`score_dmrs`
: Logical. Whether to add complementary cross-validated SVM discrimination scores to DMRs. Default is TRUE.

`extract_motifs`
: Logical. Whether to compute DMRs seeds motifs. Default is TRUE.

`bed_provided`
: Logical. Whether the beta file is provided as a BED file. Default is FALSE. In case the input has a .bed extension, this will be set to TRUE automatically.

`bed_chrom_col`
: Character. Column name for chromosome in the BED file. Default is "chrom".

`bed_start_col`
: Character. Column name for start position in the BED file. Default is "start".

`verbose`
: Numeric. Level of verbosity for logging messages, from 0 (not verbose) to 5 (very very verbose). Default is retrieved from option "CMEnt.verbose".

`.load_debug`
: Logical. If TRUE, enables debug mode for loading beta files. Default is FALSE.

Value

GRanges object of identified DMRs with metadata including DMR-level `pval` and FDR-adjusted `qval` when seed p-value aggregation is enabled.

Note on Input Data

Do not apply heavy filtering to your seeds prior to using this function, particularly based on beta values or effect sizes. The function works by expanding regions around seeds and connecting nearby sites into larger regions. Filtering out seeds with smaller effect sizes may remove important sites that could serve as "bridges" to connect more seeds into larger, biologically meaningful DMRs. For optimal results, include all statistically significant seeds (e.g., adjusted p-value < 0.05) and let the function handle region expansion and letting the function reconnect proximal sites during expansion using the ext_site_delta_beta parameter if needed. The p-value adjustment can be done using `combinePvalues()` or other methods, but avoid filtering based on beta value thresholds or effect size cutoffs before running this function. For BSSeq data, `findDMPsBSSeq()` performs seed finding using DSS.

Examples

```
beta <- matrix(
    rep(c(0.10, 0.12, 0.14, 0.78, 0.80, 0.82), each = 3), nrow = 3,
    dimnames = list(paste0("site", 1:3), paste0("S", 1:6))
)
locs <- data.frame(
    chr = "chr1", start = c(100L, 200L, 300L), row.names = rownames(beta)
)
handler <- getBetaHandler(beta, sorted_locs = locs, genome = "hg38")
```

```
pheno <- data.frame(
    Sample_Group = "cohort", casecontrol = rep(0:1, each = 3),
    row.names = colnames(beta)
)
seeds <- data.frame(pval = rep(1e-6, 3), row.names = rownames(beta))
dmrs <- buildDMRs(
    beta = handler, seeds = seeds, pheno = pheno,
    sample_group_col = "Sample_Group", casecontrol_col = "casecontrol",
    annotate_with_genes = FALSE, extract_motifs = FALSE, score_dmrs = FALSE,
    min_seeds = 2, min_sites = 2, max_lookup_dist = 150,
    testing_mode = "parametric", njobs = 1
)
```

`combinePvalues` *Combine spatially correlated p-values with comb-p style SLK correction*

Description

`combinePvalues()` is an R port of the comb-p Stouffer-Liptak-Kechris (SLK) p-value correction step. For each locus, nearby loci within `max_dist` are collected, an autocorrelation-derived covariance term is computed from their genomic distances, and the local p-values are combined with the SLK z-score formula used by comb-p.

Usage

```
combinePvalues(
  pvals,
  positions,
  chr = NULL,
  ends = positions,
  max_dist = 1000,
  lag_step = 50,
  min_dist = 0,
  lags = NULL,
  acf = NULL,
  method = c("slk", "none", stats::p.adjust.methods),
  correlation = c("spearman", "pearson", "kendall"),
  transform = TRUE,
  partial = TRUE,
  p_floor = 9e-117,
  na_correlation = 0
)
```

Arguments

`pvals`	Numeric vector of p-values.
`positions`	Numeric vector of genomic positions, one per p-value.
`chr`	Optional chromosome vector. If `NULL`, all positions are treated as coming from the same chromosome.
`ends`	Optional end positions. Defaults to `positions`, which treats each locus as a single point.

max_dist	Maximum distance used for local SLK neighborhoods and ACF estimation when acf is not supplied. Default is 1000.
lag_step	Width of distance bins used to estimate the ACF when acf is not supplied. Default is 50.
min_dist	Lower bound of the first ACF distance bin. Default is 0.
lags	Optional numeric vector of lag breakpoints. If supplied, it overrides min_dist, max_dist, and lag_step for ACF estimation.
acf	Optional precomputed ACF table with lag minimum, lag maximum, and correlation columns.
method	Either "slk"/"none" to return SLK p-values, or any method accepted by stats::p.adjust() such as "fdr"/"BH" to adjust the SLK p-values.
correlation	Correlation method used for ACF estimation. Default is "spearman", matching comb-p.
transform	Logical. If TRUE, estimate correlations on -log10(p). Default is TRUE, matching comb-p.
partial	Logical. If TRUE, estimate each lag interval from only pairs in that interval. If FALSE, outer intervals include pairs from shorter distances as in comb-p's non-partial ACF mode.
p_floor	Minimum p-value used before q-normal transformation.
na_correlation	Correlation value used for ACF bins whose correlation cannot be estimated. Default is 0.

Details

If acf is not supplied, lagged correlations are estimated from pvals and positions using comb-p's default convention of Spearman correlations on -log10(p). If acf is supplied, it should contain columns named lag_min, lag_max, and correlation, or have those values in its first three columns.

Value

A numeric vector in the same order as pvals. The estimated or supplied ACF is attached as the "acf" attribute, and the unadjusted SLK p-values are attached as "slk_pvalues" when method requests an additional multiple-testing adjustment.

References

Pedersen BS, Schwartz DA, Yang IV, Kechris KJ. Comb-p: software for combining, analyzing, grouping and correcting spatially correlated P-values. Bioinformatics. 2012;28(22):2986-2988.

Examples

```
p <- c(0.01, 0.03, 0.4, 0.02)
pos <- c(100, 150, 900, 940)
combinePvalues(p, pos, max_dist = 100)
```

computeDMRsInteraction

Compute Motif-Based DMR Interactions

Description

Computes motif-based interactions between DMRs based on their motif similarity. Identifies pairs of DMRs with significant motif similarity and returns a data frame of interactions. Assigns directionality based on score, if available.

Usage

```
computeDMRsInteraction(
  dmrs,
  genome = NULL,
  array = NULL,
  min_similarity = getOption("CMEnt.min_motif_similarity", 0.8),
  beta_locs = NULL,
  motif_site_flank_size = 5,
  find_components = TRUE,
  min_component_size = 2,
  query_components_with_jaspar = TRUE,
  plot_dir = NULL,
  output_prefix = NULL
)
```

Arguments

dmrs	Dataframe or GRanges object containing DMR coordinates and motif information
genome	Character. Genome version to use for sequence extraction. Required for data.frame DMRs; inferred from GRanges seqinfo when available.
array	Character. Array platform type (e.g., "450K", "EPIC"). Required only when beta_locs is not provided and DMR site IDs are array probe IDs.
min_similarity	Numeric. Minimum motifs PWM similarity threshold for considering DMRs are related (default: 0.8)
beta_locs	Data frame. Optional pre-computed genomic locations. If NULL, locations will be retrieved using getSortedGenomicLocs (default: NULL)
motif_site_flank_size	Integer. Number of base pairs to include as flanking regions around each site site (default: 5)
find_components	Logical. Whether to identify connected components of interacting DMRs (default: TRUE)
min_component_size	Integer. Minimum size of connected components to consider (default: 2)
query_components_with_jaspar	Logical. Whether to query connected components average PWMs against JASPAR database (default: TRUE)

plot_dir	Character. Directory to save diagnostic plots (optional). If NULL, no plots are saved (default: NULL)
output_prefix	Character. Prefix for output files to save interactions and components (optional). If NULL, results are not saved to file (default: NULL)

Value

A list with:

- interactions: Data frame of motif-based DMR interactions with columns chr1, start1, end1, chr2, start2, end2, sim, and when components are requested, component_id
- components: Data frame of discovered motif components
- dmrs: The input DMRs in the same class as provided, with an added component_ids column containing comma-separated component IDs for each DMR (or NA when the DMR is not part of a retained component)

Examples

```
# Compute motif-based interactions for DMRs
dmrs <- data.frame(
    chr = c("chr16", "chr3"),
    start = c(53468112, 37459206),
    end = c(53468712, 37493431),
    start_site = c("cg00000029", "cg00000108"),
    start_seed = c("cg00000029", "cg00000108"),
    end_site = c("cg13426503", "cg08730726"),
    end_seed = c("cg13426503", "cg08730726"),
    seeds = c("cg00000029,cg13426503", "cg00000108,cg08730726")
)
dmrs_with_motifs <- extractDMRMotifs(dmrs, genome = "hg19", array = "450K")
interactions <- computeDMRsInteraction(
    dmrs_with_motifs,
    genome = "hg19",
    array = "450K",
    query_components_with_jaspar = FALSE
)
```

---

convertBetaToTabix *Convert Beta File to Tabix-Indexed Format*

Description

Converts a methylation beta values file to a tabix-indexed BED format for faster random access during DMR analysis. The function uses a memory-efficient chunk-based approach to handle large files and can persist the derived tabix file next to analysis outputs when output_prefix is supplied.

Usage

```
convertBetaToTabix(
  beta_file,
  sorted_locs = NULL,
  array = NULL,
```

```
  genome = NULL,
  output_file = NULL,
  chunk_size = getOption("CMEnt.chunk_size", 1e+06),
  njobs = 1,
  .bed_file = NULL,
  output_prefix = NULL
)
```

Arguments

beta_file	Character. Path to the input beta values file
sorted_locs	Data frame with genomic locations containing 'chr' and 'start' columns. If NULL, will be retrieved automatically using getSortedGenomicLocs() (default: NULL)
array	Character. Array platform type. Required when sorted_locs is NULL.
genome	Character. Genome version. Required when sorted_locs is NULL.
output_file	Character. Path for the output tabix file. If NULL, a temporary file is used unless output_prefix is supplied.
chunk_size	Integer. Number of rows to process in each chunk (default: getOption("CMEnt.chunk_size", 1000000)).
njobs	Integer. Number of parallel jobs for sorting (default: 1)
.bed_file	Character. Internal precomputed BED path used to skip beta-to-BED conversion.
output_prefix	Character. Optional prefix used to persist derived tabix artifacts next to analysis outputs.

Details

The function performs the following steps:

1. Checks if tabix, bgzip, and required sorting tools are available in the system PATH
2. Processes the beta file in chunks (getOption("CMEnt.chunk_size", 1000000) rows at a time) to minimize memory usage
3. Converts beta values to BED format with genomic coordinates
4. Sorts, compresses (bgzip), and indexes (tabix) the file
5. Persists the derived file if an explicit output path or output_prefix is provided

Value

Character. Path to the created tabix file, or NULL if conversion failed

Examples

```
if (nzchar(Sys.which("tabix")) && nzchar(Sys.which("bgzip"))) {
    beta_file <- tempfile(fileext = ".tsv")
    writeLines(c("\tsample1", "cg1\t0.5"), beta_file)
    locs <- data.frame(chr = "chr1", start = 100L, row.names = "cg1")
    tabix_file <- convertBetaToTabix(beta_file, sorted_locs = locs)
}
```

`extractDMRMotifs` *Extract DMR Motif Frequencies*

Description

Extracts motif frequencies around site sites within DMRs. For each DMR, retrieves sequences around the start and end site sites, calculates base frequencies at each position, and stores the results in the DMR metadata.

Usage

```
extractDMRMotifs(
  dmrs,
  genome = NULL,
  array = NULL,
  beta_locs = NULL,
  motif_site_flank_size = 5,
  plot_dir = NULL
)
```

Arguments

`dmrs`	Dataframe or GRanges object containing DMR coordinates and site indices
`genome`	Character. Genome version to use for sequence extraction. Required for data.frame DMRs; inferred from GRanges seqinfo when available.
`array`	Character. Array platform type (e.g., "450K", "EPIC"). Required only when `beta_locs` is not provided and DMR site IDs are array probe IDs.
`beta_locs`	Data frame. Optional pre-computed genomic locations. If NULL, locations will be retrieved using getSortedGenomicLocs (default: NULL)
`motif_site_flank_size`	Integer. Number of base pairs to include as flanking regions around each site site (default: 5)
`plot_dir`	Character. Optional directory where diagnostic motif plots may be written.

Value

The input Dataframe/GRanges object with an additional metadata column:

- pwm: A matrix of base frequencies (rows: positions relative to site, columns: bases A, C, G, T)
- consensus_seq: A character string representing the consensus sequence derived from the PWM

Examples

```
# Extract motif frequencies for DMRs
dmrs <- data.frame(
    chr = c("chr16", "chr3"),
    start = c(53468112, 37459206),
    end = c(53468712, 37493431),
```

```
    start_site = c("cg00000029", "cg00000108"),
    start_seed = c("cg00000029", "cg00000108"),
    end_site = c("cg13426503", "cg08730726"),
    end_seed = c("cg13426503", "cg08730726"),
    seeds = c("cg00000029,cg13426503", "cg00000108,cg08730726")
)
dmrs_with_motifs <- extractDMRMotifs(dmrs, genome = "hg19", array = "450K")
# Access motif frequencies for the first DMR
motif_freqs_dmr1 <- dmrs_with_motifs$pwm[[1]]
```

`findDMPsArray` *Find DMPs from methylation array beta values using limma*

Description

This helper identifies differentially methylated positions (DMPs) from methylation array beta values using limma, returning the same seed-style output columns as `findDMPsBSSeq()`. In contrast to `dmpFinder()` from minfi, this function supports flexible covariate inclusion and returns a consistent output format with the BS-seq DMPs for downstream compatibility with CMEnt's region-finding functions.

Usage

```
findDMPsArray(
  beta,
  samplesheet,
  samplesheet_sep = "\t",
  sample_group_col = NULL,
  id_col = "Sample_ID",
  array = NULL,
  genome = NULL,
  sorted_locs = NULL,
  njobs = getOption("CMEnt.njobs", 1L),
  chr = "auto",
  case_group = NULL,
  covariates = NULL,
  output_file = NULL
)
```

Arguments

`beta`	A beta input supported by `getBetaHandler()`, such as a numeric matrix/data frame or a beta file path. minfi `MethylSet`/`RatioSet` inputs are also accepted and converted to beta values.
`samplesheet`	A data frame or file path to a tab-delimited sample sheet.
`samplesheet_sep`	Separator for samplesheet files. Default is tab.
`sample_group_col`	Column in `samplesheet` containing group labels.
`id_col`	Column in `samplesheet` containing sample IDs. row.names can also be used by specifying `id_col = "row.names"`.

array	Array platform passed to getSortedGenomicLocs() when sorted_locs is not supplied.
genome	Optional genome passed to getSortedGenomicLocs() when sorted_locs is not supplied. If NULL, inferred from beta when possible, otherwise from array.
sorted_locs	Optional data frame of probe locations with row names as site IDs and chr plus start or pos columns.
njobs	Number of jobs used by getBetaHandler() when reading beta files.
chr	Chromosomes to retain, "auto" for chr1-chr22, or "all" for chr1-chr22 plus chrX and chrY.
case_group	Group label to treat as case. If NULL, the first group in sample_group_col is used.
covariates	Optional covariate column names, or a comma-separated string, to include in the limma model.
output_file	Optional tab-delimited output path. Files ending in .gz are gzipped.

Value

A data frame with columns chr, start, end, site_id, pval, qval, delta_beta, and score.

Examples

```
beta <- matrix(
    c(
        0.10, 0.12, 0.11, 0.82, 0.80, 0.81,
        0.74, 0.72, 0.73, 0.20, 0.22, 0.21,
        0.40, 0.42, 0.41, 0.43, 0.44, 0.45
    ),
    nrow = 3,
    byrow = TRUE,
    dimnames = list(paste0("cg", 1:3), paste0("s", 1:6))
)
samplesheet <- data.frame(
    Sample_ID = colnames(beta),
    Sample_Group = rep(c("control", "case"), each = 3)
)
sorted_locs <- data.frame(
    chr = "chr1",
    start = c(100L, 200L, 300L),
    row.names = rownames(beta)
)
dmps <- findDMPsArray(
    beta = beta,
    samplesheet = samplesheet,
    sample_group_col = "Sample_Group",
    array = "450K",
    genome = "hg19",
    sorted_locs = sorted_locs,
    case_group = "case",
    chr = "all"
)
head(dmps)
```

`findDMPsBSSeq` *Find DMPs using DSS on BSseq objects*

Description

This helper function identifies differentially methylated positions (DMPs) from a BSseq object using the DSS package. It allows for flexible specification of sample groups, covariates, and chromosome filtering.

Usage

```
findDMPsBSSeq(
  bsseq,
  samplesheet,
  samplesheet_sep = "\t",
  sample_group_col = NULL,
  id_col = "Sample_ID",
  chr = "auto",
  case_group = NULL,
  covariates = NULL,
  output_file = NULL,
  njobs = 1L
)
```

Arguments

`bsseq`	A BSseq object or a file path to a saved BSseq object (RDS format).
`samplesheet`	A data frame or a file path to a tab-delimited text file containing sample metadata. Must include columns for sample IDs and group labels.
`samplesheet_sep`	The separator used in the samplesheet file if a file path is provided. Default is tab ("\t").
`sample_group_col`	Required name of the column in the samplesheet that contains the group labels for comparison.
`id_col`	The name of the column in the samplesheet that contains the sample IDs. Default is "Sample_ID".
`chr`	A character vector of chromosome names to include in the analysis, or "auto" to automatically include chr1-chr22, or "all" to include chr1-chr22 plus chrX and chrY. Default is "auto".
`case_group`	The specific group label in the sample_group_col to treat as the "case" group for comparison. If NULL, the first unique group in sample_group_col will be used as the case group. Default is NULL.
`covariates`	A character vector of additional covariate column names from the samplesheet to include in the DSS model, or a comma-separated string of covariate names. Default is NULL (no additional covariates).
`output_file`	An optional file path to save the DMP results as a tab-delimited text file. If the file name ends with ".gz", the output will be gzipped. Default is NULL (no file output).
`njobs`	The number of parallel jobs to use for chromosome-level analysis. Default is 1.

Value

A data frame of identified DMPs with columns for chromosome, position, site ID, p-value, q-value, delta beta, and DMP score.

Examples

```
if (requireNamespace("bsseqData", quietly = TRUE) &&
    requireNamespace("DSS", quietly = TRUE)) {
    # Load example BSseq data
    data(BS.cancer.ex, package = "bsseqData")
    BS.cancer.ex <- BS.cancer.ex[seq_len(1000), ]
    # Create a sample metadata data frame
    samplesheet <- data.frame(
       Sample_ID = colnames(BS.cancer.ex),
      Sample_Group = c(rep("Condition1", 3), rep("Condition2", 3)),
      Age = c(30, 32, 31, 28, 29, 27)
    )
    # Find DMPs with DSS
    dmps <- findDMPsBSSeq(
       bsseq = BS.cancer.ex,
       samplesheet = samplesheet,
       sample_group_col = "Sample_Group",
       id_col = "Sample_ID",
       case_group = "Condition2",
       covariates = "Age",
       output_file = NULL,
       njobs = 4
    )
    print(head(dmps))
}
```

`getBetaHandler` *Create a BetaHandler object for efficient beta value access*

Description

Create a new BetaHandler object that manages methylation beta values from various input formats (files, matrices, tabix, BSseq objects) with memory-efficient access patterns.

Usage

```
getBetaHandler(
  beta,
  array = NULL,
  genome = NULL,
  beta_row_names_file = NULL,
  sorted_locs = NULL,
  chrom_col = "#chrom",
  start_col = "start",
  output_prefix = NULL,
  njobs = getOption("CMEnt.njobs", .defaultNJobs())
)
```

Arguments

beta	Path to beta values file, or a tabix, or a beta matrix, or a BSseq object
array	Array platform type, **ignored** if sorted_locs or a BSseq object have been provided
genome	Reference genome version, eg. hg38 or hs1. Only human and mouse genomes are supported. **Ignored** if sorted_locs or a BSseq object have been provided.
beta_row_names_file	Path to row names file
sorted_locs	Data frame with genomic locations containing 'chr' and 'start' and 'end' columns, sorted by genomic position. If NULL, will be retrieved automatically using genome and array information, or extracted from BSseq object.
chrom_col	Chromosome column name in tabix file
start_col	Start position column name in tabix file
output_prefix	Prefix used for saving derived beta artifacts.
njobs	Number of parallel jobs to use when reading beta file or tabix file. Default is number of available cores minus one, up to a maximum of 8.

Value

A new BetaHandler object

Examples

```
loadExampleInputData("beta")
beta_matrix <- beta

beta_handler <- getBetaHandler(
    beta = beta_matrix,
    array = "450K",
    genome = "hg19"
)

beta_locs <- beta_handler$getBetaLocs()
head(beta_locs)

beta_values <- beta_handler$getBeta()
head(beta_values[, 1:5])
```

---

getDMRSequences *Extract DNA Sequences for DMRs*

Description

Retrieves the DNA sequences corresponding to genomic regions specified in a GRanges object. This function is useful for extracting the actual DNA sequence of identified DMRs for downstream analyses such as motif finding or sequence composition analysis.

Usage

```
getDMRSequences(
  dmrs,
  genome,
  use_online = FALSE,
  uflank_size = 0,
  dflank_size = 0,
  batch_size = 100,
  njobs = 1
)
```

Arguments

dmrs	GRanges object containing genomic coordinates of DMRs
genome	Character. Genome version to use for sequence extraction, .e.g. "hg38" or "hs1".
use_online	Logical. If TRUE, forces use of online UCSC API instead of BSgenome packages. If FALSE (default), uses BSgenome packages with online fallback when packages are unavailable (default: FALSE)
uflank_size	Integer. Number of base pairs to add as flanking regions upstream of each DMR (default: 0)
dflank_size	Integer. Number of base pairs to add as flanking regions downstream of each DMR (default: 0)
batch_size	Integer. For online API, number of regions to process per batch (default: 100)
njobs	Integer. For online API, number of cores for parallel processing (default: 1)

Details

The function first attempts to use genome-appropriate BSgenome packages:

- hg19: BSgenome.Hsapiens.UCSC.hg19
- hg38: BSgenome.Hsapiens.UCSC.hg38
- hs1: BSgenome.Hsapiens.UCSC.hs1
- mm10: BSgenome.Mmusculus.UCSC.mm10
- mm39: BSgenome.Mmusculus.UCSC.mm39

If the required BSgenome package is not installed, the function raises an error with installation instructions. Set `use_online = TRUE` to query sequences from the UCSC Genome Browser REST API instead. The online method processes sequences in batches with optional parallel processing for improved performance with large datasets.

For large numbers of DMRs (>10k), consider using parallel processing by setting njobs > 1 when using the online API, or install the appropriate BSgenome package for much faster local sequence retrieval.

Value

A Character vector containing DNA sequences for each DMR

Examples

```
dmrs <- GenomicRanges::GRanges("chr1", IRanges::IRanges(100000, 100100))
if (requireNamespace("BSgenome.Hsapiens.UCSC.hg19", quietly = TRUE)) {
    sequences <- getDMRSequences(dmrs, "hg19")
    gc_content <- vapply(sequences, function(s) {
        (stringr::str_count(s, "G") + stringr::str_count(s, "C")) / nchar(s)
    }, numeric(1))
}
```

---

getSortedGenomicLocs *Get Sorted Array Locations*

Description

Retrieves and sorts genomic location annotations for the specified methylation array platform and genome version. Performs liftOver if necessary. The function caches the results.

Usage

```
getSortedGenomicLocs(array = NULL, genome = NULL, locations_file = NULL)
```

Arguments

array	Character. Array platform type (supported: "450K", "EPIC", "EPICv2", "27K", "Mouse"), ignored when locations_file is provided.
genome	Character. Genome version (supported: "hg38", "hg19", "hs1", "mm10", "mm39"), ignored if locations_file is provided
locations_file	Character. Optional path to a precomputed locations file (RDS format). If provided, this file will be used directly (default: NULL)

Value

A data frame containing sorted genomic locations with rownames as site IDs and columns:

- chr: Chromosome
- start: Genomic position
- start: Start position (same as start)
- end: End position (start + 1)

Examples

```
locs <- data.frame(chr = c("chr1", "chr1"), start = c(100L, 200L))
rownames(locs) <- c("cg1", "cg2")
locations_file <- tempfile(fileext = ".rds")
saveRDS(locs, locations_file)

sorted_locs <- getSortedGenomicLocs(locations_file = locations_file)
head(sorted_locs)
```

`launchCMEntViewer` *Launch CMEnt Interactive Viewer*

Description

Launches a Shiny application for interactive exploration of DMR analysis results from `buildDMRs`.

Usage

```
launchCMEntViewer(
  output_prefix,
  launch_browser = TRUE,
  port = NULL,
  host = "127.0.0.1",
  diagnostic = FALSE
)
```

Arguments

`output_prefix`	Character. Prefix used when saving DMR analysis results. The function will look for files: `{output_prefix}.dmrs.tsv.gz`, `{output_prefix}.seeds_beta.tsv.gz`, and `{output_prefix}.meta.rds`.
`launch_browser`	Logical. Whether to launch in browser (default: TRUE).
`port`	Integer. Port number for Shiny server (default: auto-assigned).
`host`	Character. Host interface for the Shiny server (default: `"127.0.0.1"`). Use `"0.0.0.0"` when exposing the app from a Docker container.
`diagnostic`	Logical. Whether to enable diagnostic features for block formation visualization (default: FALSE).

Details

The viewer provides interactive access to all visualization functions in the package:

- **Overview**: Summary statistics and filterable DMR table
- **Single DMR**: Detailed view of individual DMRs with heatmap and motif logo
- **Manhattan**: Genome-wide scatter plot of DMR scores
- **Circos**: Circular genome plot with motif interactions
- **Block Formation**: Diagnostic view of DMR block detection, if `diagnostic = TRUE`

The function expects the following output files from `buildDMRs`:

- `{output_prefix}.dmrs.tsv.gz` - Main DMR results (required)
- `{output_prefix}.seeds_beta.tsv.gz` - Beta values for seeds (required)
- `{output_prefix}.meta.rds` - Viewer metadata with phenotype, array, and genome information (required)
- `{output_prefix}dmr_interactions.tsv` - DMR interactions (optional)
- `{output_prefix}dmr_components.tsv` - Motif components (optional)

Value

Invisibly returns the Shiny app object.

See Also

buildDMRs, plotDMR, plotDMRsCircos

Examples

```
if (interactive()) {
    # After running buildDMRs with output_prefix = "my_analysis"
    launchCMEntViewer(
      output_prefix = "results/my_analysis"
    )
}
```

---

loadExampleInputData *Load CMEnt Example Resources*

Description

Load one or more example resources from the **DMRsegaldata** ExperimentHub package and assign them into the caller's environment using their resource names.

Compatibility wrapper returning the chr5/chr11 example subset.

Compatibility wrapper returning the chr21/chr22 example subset.

Usage

```
loadExampleInputData(...)

loadExampleInputDataChr5And11(...)

loadExampleInputDataChr21And22(...)
```

Arguments

...	Names of the resources to load, or none to load all available resources. Available resources: • "beta": Example beta values matrix • "pheno": Example phenotype data • "dmps": Example differentially methylated positions • "array_type": Example array type annotation

Value

Invisibly returns the loaded object when a single resource is requested, or a named list of loaded objects when multiple resources are requested.

Examples

```
# Load phenotype data into the current environment
loadExampleInputData("pheno")
head(pheno)

# Load multiple resources at once
loadExampleInputDataChr5And11("beta", "dmps", "pheno", "array_type")
dim(beta)
```

`orderByLoc` *Orders a vector of indices according to their corresponding genomic locations (chromosome and position). This function is useful for sorting site sites or other genomic features by their physical positions.*

Description

Orders a vector of indices according to their corresponding genomic locations (chromosome and position). This function is useful for sorting site sites or other genomic features by their physical positions.

Usage

```
orderByLoc(x, array = NULL, genome = NULL, genomic_locs = NULL)
```

Arguments

`x`	Character or integer vector. Indices or identifiers to be ordered
`array`	Character. Array platform type, either "450K" or "EPIC". Required when `genomic_locs` is NULL.
`genome`	Character. Genome version, either "hg38", "hg19", or "hs1". Required when `genomic_locs` is NULL.
`genomic_locs`	Data frame. Optional pre-computed genomic locations. If NULL, locations will be retrieved using getSortedGenomicLocs (default: NULL)

Value

Integer vector of ordered indices

Examples

```
locs <- data.frame(chr = c("chr1", "chr1", "chr2"), start = c(100L, 200L, 50L))
rownames(locs) <- c("cg1", "cg2", "cg3")
site_ids <- c("cg2", "cg3", "cg1")

ordered_indices <- orderByLoc(site_ids, genomic_locs = locs)
site_ids[ordered_indices]
```

`plotAutoDMRsCircos` *Plot DMR Circos Views Using Automatically Selected Regions*

Description

Selects a small set of informative genomic windows from the input DMRs and forwards them to `plotDMRsCircos()`. The default `blocks` mode prefers localized high-scoring DMR blocks when `block_id` is available, while `quick` mode falls back to top-scoring individual DMR windows. Both selectors are greedy and avoid exhaustive optimization to keep region finding inexpensive.

Usage

```
plotAutoDMRsCircos(
  dmrs,
  beta,
  pheno,
  method = c("blocks", "components", "hybrid", "quick"),
  n_regions = 6,
  region_flank_bp = 1e+06,
  max_regions_per_chr = 2,
  min_inter_region_bp = 5e+06,
  genome = NULL,
  array = NULL,
  sorted_locs = NULL,
  components = NULL,
  interactions = NULL,
  min_similarity = 0.8,
  motif_site_flank_size = 5,
  min_component_size = 2,
  chromosomes = NULL,
  query_components_with_jaspar = TRUE,
  ...
)
```

Arguments

`dmrs`	GRanges object or data frame. DMR results from buildDMRs.
`beta`	BetaHandler object, character path to beta file, or beta values matrix. If not provided, beta heatmap track will be omitted.
`pheno`	Data frame or character path to phenotype file. Sample information with row-names matching beta column names (required for beta track, if not provided beta track will not be shown).
`method`	Character. Automatic region selection mode: `"blocks"` (default), `"components"`, `"hybrid"`, or `"quick"`.
`n_regions`	Integer. Target maximum number of regions to show (default: `6`).
`region_flank_bp`	Numeric. Flank in base pairs added around each selected block or DMR before plotting (default: `1e6`).

`max_regions_per_chr`
: Integer or `NULL`. Per-chromosome cap on automatically selected regions (default: `2`).

`min_inter_region_bp`
: Numeric. Nearby candidate windows within this gap are merged instead of being shown as separate Circos sectors (default: `5e6`).

`genome`
: Character. Required if DMRs are provided as a data frame or they lack seqinfo.

`array`
: Character. Array platform type (e.g., "EPIC", "450K"). Required if beta is provided as a file path and is indexed by array probe IDs.

`sorted_locs`
: Data frame. Genomic locations sorted by position (optional). If NULL, will be fetched based on array and genome.

`components`
: Data frame. Output from motif component detection (optional, will be computed if missing).

`interactions`
: Data frame. Output from motif interaction detection (optional, will be computed if missing).

`min_similarity`
: Numeric. Minimum motifs PWM similarity threshold for considering DMRs are related (default: 0.8).

`motif_site_flank_size`
: Integer. Flanking region size for motif extraction in bp (default: 5).

`min_component_size`
: Integer. Minimum motif component size to retain (default: 2).

`chromosomes`
: Character vector. Subset of chromosomes to display (default: NULL, show all available).

`query_components_with_jaspar`
: Logical. Whether computed motif components should be queried against JASPAR before plotting (default: `TRUE`). Set to `FALSE` to keep link computation cheaper.

`...`
: Additional arguments passed to `plotDMRsCircos()`. This is where plot-only settings such as `max_dmrs_per_chr`, `max_sites_per_dmr`, `max_num_samples_per_group`, `max_components`, `unmatched_interaction_color`, `legend_width_ratio`, `degenerate_resoluti` `output_file`, and `verbose` can be supplied. Arguments managed by automatic selection, including `region`, should not be passed through `...`.

Value

Invisibly returns the selected region data frame with columns `chr`, `start`, and `end`.

Examples

```
dmrs <- readRDS(system.file(
    "extdata", "example_outputChr5And11.rds", package = "CMEnt"
))
regions <- plotAutoDMRsCircos(
    dmrs, beta = NULL, pheno = NULL, method = "quick", n_regions = 1,
    query_components_with_jaspar = FALSE
)
```

plotDMR *Plot DMR*

Description

Creates a detailed DMR plot with an integrated heatmap showing beta values across samples for seeds and surrounding sites. The plot consists of two panels: the top panel shows the DMR structure with seeds and extended sites, and the bottom panel displays a heatmap of beta values for all samples, if beta values are provided. Additionally, if motif information is available or can be extracted, a sequence logo plot is added showing the nucleotide composition and information content around site sites in the DMR.

Usage

```
plotDMR(
  dmrs,
  dmr_index,
  beta = NULL,
  pheno = NULL,
  genome = NULL,
  array = NULL,
  beta_locs = NULL,
  sample_group_col = NULL,
  extend_by_dmr_size_ratio = 0.2,
  min_extension_bp = 50,
  max_sites = 100,
  max_samples_per_group = 10,
  plot_motif = TRUE,
  motif_site_flank_size = 5,
  plot_title = TRUE,
  output_file = NULL,
  width = 8,
  height = 12
)
```

Arguments

dmrs	GRanges object. Output from buildDMRs.
dmr_index	Integer. Which DMR to plot.
beta	BetaHandler object, character path to beta file, or beta values matrix. If a character path or matrix is provided, a BetaHandler will be created automatically.
pheno	Data frame or character path to phenotype file. Sample information with rownames matching beta column names (required).
genome	Character. Genome version. Required for data.frame DMRs; inferred from GRanges seqinfo when available.
array	Character. Array platform type. Required when beta row names are array probe IDs and beta_locs is not provided.
beta_locs	Data frame. Genomic locations sorted by position (optional).

`sample_group_col`
: Character. Required column in pheno for sample grouping when pheno is provided.

`extend_by_dmr_size_ratio`
: Numeric. Ratio of the DMR width to extend the plot region outside of the DMR in both sides (default: 0.2).

`min_extension_bp`
: Integer. Minimum extension in base pairs (default: 50).

`max_sites`
: Integer. Maximum number of sites to show in heatmap (default: 100).

`max_samples_per_group`
: Integer. Maximum number of samples to show per group in heatmap (default: 10).

`plot_motif`
: Logical. Whether to plot the sequence logo motif (default: TRUE).

`motif_site_flank_size`
: Integer. Number of base pairs to include as flanking regions around each site site for motif extraction (default: 5).

`plot_title`
: Logical. Whether to display the title on the plot. If FALSE, the title is shown in the logs (default: TRUE).

`output_file`
: Character. If provided, saves the plot to the specified file path (PDF format).

`width`
: Numeric. Width of the output PDF in inches when `output_file` is provided (default: `8`).

`height`
: Numeric. Height of the output PDF in inches when `output_file` is provided (default: `12`).

Value

A combined plot object (gridExtra) containing the DMR structure plot, beta values heatmap (if beta is provided), and sequence logo motif plot (if motif information is available and plot_motif is TRUE).

Examples

```
dmrs <- readRDS(system.file(
    "extdata", "example_outputChr5And11.rds", package = "CMEnt"
))
if (requireNamespace(
    "IlluminaHumanMethylation450kanno.ilmn12.hg19", quietly = TRUE
)) {
    plotDMR(dmrs, 1, array = "450K", plot_motif = FALSE)
}
```

---

plotDMRBlockFormation *Plot Intermediate DMR Block Formation Diagnostics*

Description

Visualizes how DMR blocks are formed on a single chromosome by layering raw scores, smoothed scores, piecewise-linear segments, slope-based candidate blocks, distance-based split boundaries, and final accepted blocks.

Usage

```
plotDMRBlockFormation(
  dmrs,
  chromosome,
  genome = NULL,
  k_neighbors = 5L,
  min_segment_size = 2L,
  block_gap_mode = "adaptive",
  block_gap_fixed_bp = NULL,
  block_gap_quantile = 0.95,
  block_gap_multiplier = 1.5,
  block_gap_min_bp = 250000,
  block_gap_max_bp = 5e+06,
  point_alpha = 0.7,
  point_size = 1
)
```

Arguments

dmrs	GRanges object or data frame. DMR results from buildDMRs or scoreDMRs.
chromosome	Character. Chromosome to inspect (e.g., "chr7" or "7").
genome	Character. Genome version. Required for data.frame DMRs; inferred from GRanges seqinfo when available.
k_neighbors	Integer. Number of nearest neighbors used in adaptive Gaussian smoothing (default: 5).
min_segment_size	Integer. Minimum size of linear segments in PELT segmentation (default: 2).
block_gap_mode	Character. Gap rule for block splitting: "adaptive" (default), "fixed", or "none".
block_gap_fixed_bp	Numeric. Maximum allowed midpoint gap (bp) when block_gap_mode = "fixed". Ignored otherwise.
block_gap_quantile	Numeric in (0, 1). Gap quantile used for adaptive thresholding (default: 0.95).
block_gap_multiplier	Numeric > 0. Multiplier for adaptive gap threshold (default: 1.5).
block_gap_min_bp	Numeric >= 0. Lower clamp for adaptive gap threshold (bp). Default is 250000.
block_gap_max_bp	Numeric >= block_gap_min_bp. Upper clamp for adaptive gap threshold (bp). Default is 5000000.
point_alpha	Numeric. Alpha for raw score points in [0, 1] (default: 0.7).
point_size	Numeric. Point size for raw scores (default: 1.0).

Value

A ggplot object.

Examples

```
dmrs <- data.frame(
    chr = "chr7",
    start = seq(1e6, by = 5e4, length.out = 10),
    end = seq(1e6, by = 5e4, length.out = 10) + 100,
    score = seq(0.5, 0.8, length.out = 10)
)
p <- plotDMRBlockFormation(dmrs, chromosome = "chr7", genome = "hg38")
```

---

plotDMRs *Plot Multiple DMRs in a Grid*

Description

Creates a grid of DMR plots for multiple regions. Can optionally include beta value heatmaps if beta and pheno data are provided.

Usage

```
plotDMRs(
  dmrs,
  dmr_indices = NULL,
  top_n = 4,
  score_by = c("delta_beta", "score"),
  beta = NULL,
  pheno = NULL,
  sample_group_col = NULL,
  genome = NULL,
  array = NULL,
  beta_locs = NULL,
  ncol = 1,
  output_file = NULL,
  width = 8,
  height = 12,
  ...
)
```

Arguments

dmrs	GRanges object. Output from buildDMRs.
dmr_indices	Integer vector. Which DMRs to plot. If NULL, selects top_n DMRs using score_by.
top_n	Integer. Number of top DMRs to plot when dmr_indices is NULL (default: 4).
score_by	Character. Which metric to use for ranking DMRs when dmr_indices is NULL. Options: "delta_beta" or "score" (default: "delta_beta").
beta	BetaHandler object, character path to beta file, or beta values matrix (optional). If provided, creates plots with heatmaps.
pheno	Data frame or character path to phenotype file (optional). Required when beta is provided.

sample_group_col
: Character. Required column in pheno for sample grouping when pheno is provided.

genome
: Character. Genome version. Required for data.frame DMRs; inferred from GRanges seqinfo when available.

array
: Character. Array platform type. Required when beta row names are array probe IDs and beta_locs is not provided.

beta_locs
: Data frame. Genomic locations sorted by position (optional). If NULL, will be fetched based on array and genome.

ncol
: Integer. Number of columns in the grid (default: 1).

output_file
: Character. Path to save the combined plot as PDF (optional). If NULL, the plot is not saved to file.

width
: Numeric. Width of the output PDF in inches (default: 8).

height
: Numeric. Height of the output PDF in inches (default: 12).

...
: Additional arguments passed to plotDMR.

Value

If beta is NULL: A gtable object. If beta is provided: A list of combined plot objects with structure and heatmap.

Examples

```
dmrs <- readRDS(system.file(
    "extdata", "example_outputChr5And11.rds", package = "CMEnt"
))
if (requireNamespace(
    "IlluminaHumanMethylation450kanno.ilmn12.hg19", quietly = TRUE
)) {
    plotDMRs(dmrs, dmr_indices = seq_len(min(2, length(dmrs))),
             array = "450K", ncol = 2, plot_motif = FALSE)
}
```

---

plotDMRsCircos *Plot Circos Visualization of DMRs*

Description

Creates a circular genome plot using circlize showing DMRs with multiple tracks: ideogram track with chromosome bands, DMR arcs colored by delta beta, beta value heatmaps for each sample, and motif-based interaction links between DMRs.

Usage

```
plotDMRsCircos(
  dmrs,
  beta = NULL,
  pheno = NULL,
  genome = NULL,
```

```
  array = NULL,
  sorted_locs = NULL,
  components = NULL,
  interactions = NULL,
  sample_group_col = NULL,
  min_similarity = 0.8,
  motif_site_flank_size = 5,
  max_num_samples_per_group = 5,
  max_dmrs_per_chr = 10,
  max_sites_per_dmr = 5,
  min_component_size = 2,
  max_components = 30,
  chromosomes = NULL,
  region = NULL,
  query_components_with_jaspar = TRUE,
  unmatched_interaction_color = "#B3B3B3",
  group_colors = NULL,
  low_beta_color = "#2b83ba",
  mid_beta_color = "#f7f7f7",
  high_beta_color = "#d7191c",
  neg_delta_color = "#055709",
  zero_delta_color = "#f0ec10",
  pos_delta_color = "#801414",
  legend_width_ratio = 0.5,
  degenerate_resolution = 1e+06,
  output_file = NULL,
  verbose = NULL
)
```

Arguments

dmrs	GRanges object or data frame. DMR results from buildDMRs.
beta	BetaHandler object, character path to beta file, or beta values matrix. If not provided, beta heatmap track will be omitted.
pheno	Data frame or character path to phenotype file. Sample information with rownames matching beta column names (required for beta track, if not provided beta track will not be shown).
genome	Character. Required if DMRs are provided as a data frame or they lack seqinfo.
array	Character. Array platform type (e.g., "EPIC", "450K"). Required if beta is provided as a file path and is indexed by array probe IDs.
sorted_locs	Data frame. Genomic locations sorted by position (optional). If NULL, will be fetched based on array and genome.
components	Data frame. Output from motif component detection (optional, will be computed if missing).
interactions	Data frame. Output from motif interaction detection (optional, will be computed if missing).
sample_group_col	Character. Required column in pheno for sample grouping when pheno is provided.
min_similarity	Numeric. Minimum motifs PWM similarity threshold for considering DMRs are related (default: 0.8).

`motif_site_flank_size`
: Integer. Flanking region size for motif extraction in bp (default: 5).

`max_num_samples_per_group`
: Integer. Maximum number of samples to show per group in heatmap (default: 5).

`max_dmrs_per_chr`
: Integer. Maximum number of DMRs to use per chromosome (default: 10). The DMRs with highest absolute delta beta will be selected.

`max_sites_per_dmr`
: Integer. Maximum number of sites to show per DMR in scatter/heatmap (default: 5).

`min_component_size`
: Integer. Minimum motif component size to retain (default: 2).

`max_components`
: Integer. Maximum number of interactions to plot (default: 30).

`chromosomes`
: Character vector. Subset of chromosomes to display (default: NULL, show all available).

`region`
: Genomic region to display. Can be NULL, a GRanges, a string in the form `chr:start-end`, or a data.frame/list with columns `chr`, `start`, `end`.

`query_components_with_jaspar`
: Logical. Whether computed motif components should be queried against JASPAR before plotting (default: `TRUE`). Set to `FALSE` to keep link computation cheaper.

`unmatched_interaction_color`
: Character. Color used for interaction components without JASPAR matches. These links are shown but omitted from the interaction legend (default: `"#B3B3B3"`).

`group_colors`
: Named character vector of colors for sample groups labels in the beta values track (names should match unique values in `pheno[[sample_group_col]]`). If not provided, default colors will be used.

`low_beta_color`
: Character. Color for low beta values in the heatmap (default: "#2b83ba").

`mid_beta_color`
: Character. Color for mid beta values in the heatmap (default: "#f7f7f7").

`high_beta_color`
: Character. Color for high beta values in the heatmap (default: "#d7191c").

`neg_delta_color`
: Character. Color for negative delta beta values in the DMR arcs (default: "#055709").

`zero_delta_color`
: Character. Color for zero delta beta values in the DMR arcs (default: "#f7f7f7").

`pos_delta_color`
: Character. Color for positive delta beta values in the DMR arcs (default: "#801414").

`legend_width_ratio`
: Numeric. Fraction of horizontal canvas reserved for legends (default: 0.50).

`degenerate_resolution`
: Integer. Resolution in base pairs for simplifying narrow glyphs: link ribbons are drawn as lines when both anchors are below this span, and DMR arcs are drawn as lines instead of rectangles below this span (default: 1e6).

`output_file`
: Character or NULL. Optional PDF path for the plot.

`verbose`
: Numeric. Optional verbosity override.

Value

NULL invisibly after drawing the plot.

Examples

```
dmrs <- readRDS(system.file(
    "extdata", "example_outputChr5And11.rds", package = "CMEnt"
))
plotDMRsCircos(dmrs, query_components_with_jaspar = FALSE)
```

---

plotDMRsManhattan *Plot Manhattan-Style View of DMR Scores*

Description

Creates a Manhattan-style genome-wide scatter plot of DMR scores. DMRs are colored by their dominant genomic region class inferred from DMR annotations.

Usage

```
plotDMRsManhattan(
  dmrs,
  region = NULL,
  genome = NULL,
  promoter_col = "in_promoter_of",
  gene_body_col = "in_gene_body_of",
  point_size = 1.1,
  point_alpha = 0.75,
  block_col = "block_id",
  show_blocks = TRUE,
  block_alpha = 0.12,
  block_linewidth = 0.25,
  output_file = NULL,
  add_hover_text = FALSE,
  width = 12,
  height = 6
)
```

Arguments

dmrs	GRanges object or data frame. DMR results from buildDMRs.
region	Optional plotting scope. Can be NULL for full-chromosome plotting, a GRanges, a string in the form "chr:start-end", or a data.frame/list with chr, start, and end.
genome	Character. Genome version. Required for data.frame DMRs; inferred from GRanges seqinfo when available.
promoter_col	Character. Metadata column indicating promoter overlap (default: "in_promoter_of").
gene_body_col	Character. Metadata column indicating gene-body overlap (default: "in_gene_body_of").
point_size	Numeric. Point size (default: 1.1).

point_alpha	Numeric. Point alpha in [0, 1] (default: 0.75).
block_col	Character. Metadata column containing block IDs (default: "block_id").
show_blocks	Logical. If TRUE, draw translucent rectangles for identified blocks (default: TRUE).
block_alpha	Numeric. Alpha for block rectangles in [0, 1] (default: 0.12).
block_linewidth	Numeric. Line width for block rectangle borders (default: 0.25).
output_file	Character or NULL. If non-NULL, path to save the plot as a PDF (default: NULL).
add_hover_text	Logical. Whether to add plotly-compatible hover text to points and block rectangles (default: FALSE).
width	Numeric. Width of the output PDF in inches (default: 12).
height	Numeric. Height of the output PDF in inches (default: 6).

Value

A ggplot object.

Examples

```
dmrs <- data.frame(
    chr = c("chr1", "chr1", "chr2"),
    start = c(100, 200, 100),
    end = c(150, 250, 150),
    score = c(0.6, 0.8, 0.7)
)
p <- plotDMRsManhattan(dmrs, genome = "hg38")
```

---

readCustomMethylationBedData

Read and Process Custom Methylation BED Data

Description

Reads methylation data from a custom BED file format, converts it to a tabix-indexed format for efficient random access, and creates genomic location indices. This function is designed to handle custom methylation array data or sequencing-based methylation data in BED format, making it compatible with the CMEnt workflow.

Usage

```
readCustomMethylationBedData(
  bed_file,
  pheno,
  genome,
  chrom_col = "#chrom",
  start_col = "start",
  output_dir = NULL,
  chunk_size = getOption("CMEnt.chunk_size", 1e+06),
  output_prefix = NULL,
  njobs = 1L
)
```

Arguments

bed_file	Character. Path to the input BED file containing methylation data. The file should have chromosome and position columns, plus sample columns with methylation values. Can be gzipped (default: NULL)
pheno	Data frame. Phenotype data with sample IDs as rownames. Only samples present in both the pheno rownames and BED file header will be processed
genome	Character. Required genome version to use (e.g., "hg38", "hg19", "hs1").
chrom_col	Character. Name of the chromosome column in the BED file (default: "#chrom")
start_col	Character. Name of the start position column in the BED file (default: "start")
output_dir	Character. Directory for caching processed files. If NULL, uses a temporary working directory unless output_prefix is provided (default: NULL)
chunk_size	Integer. Number of rows to process in each chunk for memory efficiency (default: getOption("CMEnt.chunk_size", 1000000)). This allows processing of large BED files without loading the entire dataset into memory.
output_prefix	Character. Optional prefix used to persist derived BED/tabix artifacts next to analysis outputs.
njobs	Integer. Number of jobs to use after normalization. On Unix-like systems, location-registry creation and tabix conversion run in parallel when njobs > 1.

Details

The function performs the following workflow:

1. Validates that tabix and bgzip are available in the system PATH
2. Checks the BED file header for required columns and sample IDs
3. Processes the BED file in chunks to minimize memory usage
4. Normalizes the BED format with standard BED6 columns (#chrom, start, end, id, score, strand)
5. Converts chromosomes to integer factors for efficient sorting
6. Creates a tabix-indexed compressed file for fast random access
7. Persists derived artifacts under output_prefix when provided

Value

A list with two elements:

- tabix_file: Character path to the created tabix-indexed BED file
- locations: Disk-backed genomic location registry

Requirements

This function requires tabix and bgzip command-line tools to be installed and available in the system PATH. These tools are part of the HTSlib/samtools suite.

Memory Management

The function uses chunk-based processing to handle large BED files without loading the entire dataset into memory. The genomic locations are stored in a Registry object that can exceed available RAM by using disk-backed storage.

See Also

convertBetaToTabix for converting standard beta files to tabix format getBetaHandler for creating a BetaHandler object from processed files

Examples

```
# Create a simple phenotype data frame
pheno <- data.frame(
    sample_group = c("case", "control"),
    row.names = c("Sample1", "Sample2")
)

if (nzchar(Sys.which("tabix")) && nzchar(Sys.which("bgzip"))) {
    bed_file <- tempfile(fileext = ".bed")
    writeLines(c(
        "#chrom\tstart\tSample1\tSample2",
        "chr1\t100\t0.2\t0.8",
        "chr1\t200\t0.3\t0.7"
    ), bed_file)
    result <- readCustomMethylationBedData(bed_file, pheno)
    result$tabix_file
}
```

---

reduceBSseqAroundSeeds

Reduce BSseq object around seed locations

Description

This function reduces a BSseq object to only include methylation data around specified seed locations. It expands the seed locations by a given window size and merges overlapping windows to create a reduced BSseq object.

Usage

```
reduceBSseqAroundSeeds(bsseq_obj, seeds, expansion_window)
```

Arguments

bsseq_obj	BSseq object containing the methylation data
seeds	Data frame with genomic locations containing 'chr' and 'start' columns.
expansion_window	Integer. DMR maximum stage 2 bi-directional size expansion window (default: 10000).

Details

The reduction is performed by expanding the seed locations by the specified window size to both sides, after division by 2, and merging overlapping windows.The resulting BSseq object contains only the methylation data within these merged windows.

Value

BSseq object containing the reduced methylation data

Examples

```
library(bsseq)
# Example BSseq object
bsseq_obj <- BSseq(
    chr = c("chr1", "chr1", "chr2"),
    pos = c(100, 200, 300),
    M = matrix(c(10, 20, 30), ncol = 1),
    Cov = matrix(c(100, 200, 300), ncol = 1)
)
# Example seeds data frame
seeds <- data.frame(chr = c("chr1", "chr2"), start = c(80, 250))
reduced_bsseq_obj <- reduceBSseqAroundSeeds(bsseq_obj, seeds, expansion_window = 50)
```

---

scoreDMRs *Add Complementary Classification Scores to DMRs*

Description

Scores Differentially Methylated Regions (DMRs) based on their ability to discriminate between sample groups using cross-validated Support Vector Machine (SVM) classification. For each DMR, this function performs stratified k-fold cross-prediction using an RBF kernel SVM and computes a margin-sensitive classification score based on decision values, which serves as a complementary measure of the DMR's discriminative power. Use this score alongside DMR-level `pval`, `qval`, and effect-size columns rather than as a replacement for statistical evidence. The scores are then smoothed along the genome using a Gaussian-kNN approach, and piecewise-linear segments are detected using the PELT algorithm, expecting a rising->plateau->decreasing pattern. Finally, DMRs are assigned to localized blocks based on the smoothed score profiles and specified gap rules.

Usage

```
scoreDMRs(
  dmrs,
  beta,
  pheno,
  covariates = NULL,
  genome = NULL,
  array = NULL,
  sorted_locs = NULL,
  sample_group_col = NULL,
  block_gap_mode = c("adaptive", "fixed", "none"),
  block_gap_fixed_bp = NULL,
  block_gap_quantile = 0.95,
  block_gap_multiplier = 1.5,
  block_gap_min_bp = 250000,
  block_gap_max_bp = 5e+06,
  njobs = getOption("CMEnt.njobs", .defaultNJobs()),
  show_progress = TRUE,
```

```
  verbose = getOption("CMEnt.verbose", 1L),
  .dmr_beta = NULL,
  .memory_njobs = njobs
)
```

Arguments

dmrs	Data frame or GRanges object containing DMR coordinates and metadata
beta	Character. Path to beta value file, tabix file, beta matrix, BetaHandler object, or bed file
pheno	Data frame. Phenotype data containing sample group information
covariates	Character vector of covariate columns in pheno to regress out before scoring. Default is NULL.
genome	Character. Genome version (e.g., "hg38", "hg19", "hs1", "mm10"). Required if DMRs are provided as a data frame without seqlevels style information. Default is NULL.
array	Character. Array platform type (e.g., "450K", "EPIC", "EPICv2"). Required when beta row names are array probe IDs and sorted_locs is not provided.
sorted_locs	Data frame. Optional pre-computed sorted genomic locations. Default is NULL
sample_group_col	Character. Required column name in pheno containing sample group information.
block_gap_mode	Character. Distance rule for block construction: "adaptive" (default), "fixed", or "none".
block_gap_fixed_bp	Numeric. Maximum allowed midpoint gap (bp) when block_gap_mode = "fixed". Ignored otherwise.
block_gap_quantile	Numeric in (0, 1). Quantile of chromosome DMR midpoint gaps used in adaptive thresholding. Default is 0.95.
block_gap_multiplier	Numeric > 0. Multiplier applied to the adaptive gap quantile. Default is 1.5.
block_gap_min_bp	Numeric >= 0. Lower clamp for adaptive gap threshold (bp). Default is 250000.
block_gap_max_bp	Numeric >= block_gap_min_bp. Upper clamp for adaptive gap threshold (bp). Default is 5000000.
njobs	Integer. Number of parallel jobs used for cross-validated scoring. Default comes from getOption("CMEnt.njobs").
show_progress	Logical. Whether to display a progress bar during scoring. Default is TRUE.
verbose	Numeric. Logging verbosity level. Default comes from getOption("CMEnt.verbose").
.dmr_beta	Internal optional matrix-like object of preloaded beta values for all DMR sites. When provided, it must contain all sites referenced by mcols(dmrs)$sites and all beta sample columns.
.memory_njobs	Internal number of concurrently active jobs to use for RAM-derived materialization chunk sizing. Defaults to njobs.

Details

The function uses stratified k-fold cross-prediction to ensure balanced representation of sample groups in each fold. The number of folds can be controlled using the option "CMEnt.scoring_nfold" (default is 5). An RBF (Radial Basis Function) kernel SVM is trained on the beta values of site sites within each DMR. For reproducible fold assignments, call `set.seed()` before `scoreDMRs()`.

The `score` combines classification correctness and margin confidence, making it more sensitive than plain cross-validated accuracy when many DMRs classify perfectly. It is a complementary ranking and diagnostic measure, especially useful for sample-level separation. The `cv_accuracy` column stores the raw cross-validated accuracy for reference. Blocks are detected from smoothed score profiles and split at large midpoint gaps using the selected `block_gap_mode`.

Value

GRanges object with DMRs ordered by complementary classification score and additional metadata columns:

- score: Margin-sensitive cross-validated classification score for the DMR
- cv_accuracy: Raw cross-validated classification accuracy for the DMR
- score_smoothed: Gaussian-kNN smoothed score trajectory per chromosome
- segment_id: Piecewise-linear segment index estimated with PELT
- segment_slope: Estimated slope of the segment that each DMR belongs to
- block_id: Localized DMR block label (NA for DMRs not assigned to a block)

Examples

```
# Load example data
loadExampleInputDataChr5And11()

# Load pre-computed DMRs
dmrs <- readRDS(system.file("extdata", "example_outputChr5And11.rds", package = "CMEnt"))

# score DMRs
scoring_dmrs <- scoreDMRs(
    dmrs = dmrs[1],
    beta = beta,
    pheno = pheno,
    sample_group_col = "Sample_Group",
    array = "450K"
)
```

---

simulateDMRs *Simulate DMRs while preserving local methylation structure*

Description

This simulator is inspired by `dmrseq::simDMRs()`, but adds differential signal on the logit methylation scale. The same regional perturbation engine is used across methylation assays by operating on methylated-count and coverage-count representations. For microarray-style beta values, pseudo counts are constructed first, then the same simulation mechanism is applied.

Usage

```
simulateDMRs(
  beta,
  num_dmrs = 3000L,
  delta_max0 = 0.4,
  groups = NULL,
  case_group = NULL,
  samplesheet = NULL,
  samplesheet_sep = "\t",
  sample_group_col,
  covariates = NULL,
  max_gap = 500L,
  min_sites = 5L,
  max_sites = 500L,
  min_dmr_size = 50L,
  max_dmr_size = 2000L,
  delta_jitter = 1/3,
  expected_correlation = 0.7,
  neighbor_window = 5L,
  profile_degree = 4L,
  flank_fraction = 0.2,
  resample_counts = TRUE,
  array = NULL,
  genome = NULL,
  sorted_locs = NULL,
  beta_row_names_file = NULL,
  chrom_col = "#chrom",
  start_col = "start",
  njobs = getOption("CMEnt.njobs", .defaultNJobs()),
  verbose = getOption("CMEnt.verbose", 1)
)
```

Arguments

`beta`	A BSseq object, a BetaHandler object, a beta matrix/data frame, a path to a beta/tabix file, or a path to an `.rds` file containing a BSseq object.
`num_dmrs`	Number of DMRs to spike in.
`delta_max0`	Baseline maximum beta-scale effect size near the center of each DMR.
`groups`	Optional sample group vector. If `NULL`, the first half of samples are assigned to `Condition1` and the remaining samples to `Condition2`.
`case_group`	Group receiving the differential shift. Defaults to the second group level.
`samplesheet`	Optional data frame or path to a tab-delimited sample sheet with covariates. Row names, a `Sample_ID` column, or an unnamed first column must identify samples. If fewer than all input samples are present, simulation is run on the matching samples.
`samplesheet_sep`	Separator for samplesheet files. Default is tab.
`sample_group_col`	Required name of the phenotype column updated in the returned samplesheet. Values are `"case"` for perturbed samples and `"control"` for all other samples.

If the column already exists in a supplied samplesheet and has more than one value, the original values are retained in a separate covariate column before simulation labels are written.

`covariates` Optional covariate columns in `samplesheet` to regress out on the logit methylation scale before DMRs are simulated. The per-site baseline is retained.

`max_gap` Maximum gap, in bp, used to form candidate site segments.

`min_sites` Minimum number of sites per candidate DMR segment.

`max_sites` Maximum number of sites per candidate DMR segment.

`min_dmr_size` Minimum genomic width, in bp, per candidate DMR segment.

`max_dmr_size` Maximum genomic width, in bp, per candidate DMR segment.

`delta_jitter` Width of the random effect-size jitter around `delta_max0`.

`expected_correlation`
Minimum within-DMR expected correlation target. The per-DMR target is the larger of this value and the estimated local background correlation, capped to the interval `[0, 0.99]`.

`neighbor_window`
Number of neighboring sites used to smooth the index-based correlated random effect inside each DMR. A value of `5` uses up to two upstream and two downstream neighboring sites.

`profile_degree` Degree used by the triweight profile.

`flank_fraction` Fraction of the selected segment width added on both sides before evaluating the triweight profile.

`resample_counts`
If `TRUE`, methylated counts in touched regions are redrawn from shifted probabilities and coverage. If `FALSE`, counts are rounded deterministically.

`array` Array platform type. Used only when `beta` is not a BSseq object or a self-contained BetaHandler.

`genome` Reference genome. Used only when `beta` is not a BSseq object or a self-contained BetaHandler.

`sorted_locs` Optional genomic locations with `chr`, `start`, and optionally `end` columns. Used only for non-BSseq inputs.

`beta_row_names_file`
Optional file with beta row names. Used only for non-BSseq file inputs.

`chrom_col` Chromosome column name for tabix inputs.

`start_col` Start column name for tabix inputs.

`njobs` Number of parallel jobs for loading non-BSseq inputs.

`verbose` Numeric. Logging verbosity level. Default comes from `getOption("CMEnt.verbose")`.

Details

All inputs are first represented as methylated counts `M` and coverage counts `C`, with beta values estimated as `(M + 0.5) / (C + 1)`. Non-BSseq beta matrices are converted to pseudo-counts with fixed coverage 100. If `covariates` are supplied, their linear effects are removed on the logit methylation scale while preserving each site's mean signal.

Candidate regions are contiguous genomic site segments: adjacent sites remain in the same segment when they are on the same chromosome and no more than `max_gap` bp apart. Eligible segments have between `min_sites` and `max_sites` sites and between `min_dmr_size` and `max_dmr_size`

bp. Segment sampling is mildly weighted toward regions whose median methylation is near 0.5, reducing floor and ceiling effects.

For each selected DMR, a smooth regional effect profile is evaluated over the selected positions after expanding the segment by `flank_fraction` on both sides. With center `c`, expanded width `W`, and `d = profile_degree`, the site weight is `w_i = (1 - min(|pos_i - c| / (W / 2), 1)^d)^d`. A signed maximum effect is then drawn as `s * delta_max * w_i`, where `s` is chosen as hyper- or hypomethylating and is flipped when needed to avoid the beta-scale boundary.

Case-sample probabilities are shifted through the logit scale. For baseline probability `p_i`, intended beta-scale shift `a_i`, and correlated site-level noise `e_i`, the simulated case probability is `plogis(qlogis(p_i) + qlogis(clamp(p_i + a_i)) - qlogis(p_i) + e_i)`. The noise term is generated by smoothing independent normal variates across neighboring sites defined by `neighbor_window`. Its standard deviation is calibrated from a small lookup table so that the induced adjacent-site correlation approaches the larger of `expected_correlation` and the fitted local background correlation, capped at 0.99.

Finally, touched methylated counts are regenerated from the shifted probabilities and original coverage. With `resample_counts = TRUE`, counts are drawn as `Binomial(C, p)`; otherwise they are deterministically rounded as `round(C * p)`.

Value

A list with simulated output (`simulated`), optional genomic locations for non-BSseq inputs (`beta_locs`), phenotype data (`pheno`), and dmrseq-like metadata: `gr.dmrs`, `dmr.mncov`, `dmr.L`, `delta`, `truth`, `selected_regions`, `groups`, and `case_group`. For reproducible simulations, call `set.seed()` before `simulateDMRs()`.

Examples

```
beta <- matrix(seq(0.1, 0.9, length.out = 48), nrow = 12)
rownames(beta) <- paste0("cg", seq_len(nrow(beta)))
colnames(beta) <- paste0("sample", seq_len(ncol(beta)))
locs <- data.frame(
    chr = "chr1",
    start = seq(100, by = 100, length.out = nrow(beta)),
    row.names = rownames(beta)
)
set.seed(1)
simulated <- simulateDMRs(
    beta, num_dmrs = 1, min_sites = 5, max_sites = 20,
    sorted_locs = locs, sample_group_col = "Sample_Group", verbose = 0
)
```

---

sortBetaFileByCoordinates

Sort Beta File by Genomic Coordinates

Description

This helper function sorts a methylation beta values file by genomic coordinates (chromosome and position) as required by the buildDMRs function. The function reads the beta file, sorts the site sites according to their genomic positions using array annotation, and writes the sorted data to a new file.

Usage

```
sortBetaFileByCoordinates(
  beta_file,
  output_file = NULL,
  array = NULL,
  genome = NULL,
  genomic_locs = NULL,
  overwrite = FALSE
)
```

Arguments

beta_file	Character. Path to the input beta values file to be sorted
output_file	Character. Path for the output sorted beta file (default: adds "_sorted" suffix)
array	Character. Array platform type. Required when genomic_locs is NULL.
genome	Character. Genome version. Required when genomic_locs is NULL.
genomic_locs	Data frame. Optional pre-computed genomic locations. If NULL, locations will be retrieved automatically (default: NULL)
overwrite	Logical. Whether to overwrite existing output file (default: FALSE)

Details

The function performs the following steps:

1. Reads the beta values file
2. Loads the appropriate array annotation (450K or EPIC)
3. Sorts site sites by genomic coordinates (chr:start)
4. Writes the sorted data to a new file
5. Validates that the output is properly sorted

Value

Character. Path to the sorted output file

Note

If you want to convert to tabix, consider using the convertBetaToTabix function instead directly, sorting is done internally.

Examples

```
beta_file <- tempfile(fileext = ".tsv")
writeLines(c("sample1", "cg2\t0.2", "cg1\t0.1"), beta_file)
locs <- data.frame(
    chr = c("chr1", "chr1"),
    start = c(100L, 200L),
    row.names = c("cg1", "cg2")
)
sorted_file <- sortBetaFileByCoordinates(beta_file, genomic_locs = locs)
```

`sparsifySeeds` *Sparsify seed locations by minimum distance*

Description

Removes seeds inside dense runs where neighboring seeds are closer than a minimum distance on the same chromosome.

Usage

```
sparsifySeeds(seeds, min_distance)
```

Arguments

`seeds`	Data frame with genomic seed locations containing `chr` and `start` columns.
`min_distance`	Non-negative numeric scalar. Seeds with a distance below this value to both neighboring seeds on the same chromosome are removed.

Details

Seeds are compared per chromosome after sorting by `start`. A seed is removed only when both the previous and next seed on that chromosome are closer than `min_distance`. Endpoints of close runs are retained. Seeds at exactly `min_distance` are retained.

Value

A subset of `seeds` with original columns and row names preserved.

Examples

```
seeds <- data.frame(
    chr = c("chr1", "chr1", "chr1"),
    start = c(100, 105, 120)
)
sparsifySeeds(seeds, min_distance = 10)
```

Index

∗ **internal**
    BetaHandler, 8
    CMEnt-package, 3

annotateDMRsWithGenes, 5
augmentBSSeq, 6

BetaHandler, 8, *47, 48*
buildDMRs, *3*, 11, *28, 29*

CMEnt *(*CMEnt-package*)*, 3
CMEnt-package, 3
combinePvalues, 15
combinePvalues(), *14*
computeDMRsInteraction, *3*, 17
convertBetaToTabix, 18, *43*

dmpFinder(), *21*

extractDMRMotifs, *3*, 20

findDMPsArray, 21
findDMPsBSSeq, 23
findDMPsBSSeq(), *14*, *21*

getBetaHandler, 24, *43*
getBetaHandler(), *21, 22*
getDMRSequences, 25
getSortedGenomicLocs, 27
getSortedGenomicLocs(), *22*

launchCMEntViewer, 28
loadExampleInputData, 29
loadExampleInputDataChr21And22
        *(*loadExampleInputData*)*, 29
loadExampleInputDataChr5And11
        *(*loadExampleInputData*)*, 29

orderByLoc, 30

plotAutoDMRsCircos, 31
plotDMR, *3*, *29*, 33
plotDMRBlockFormation, 34
plotDMRs, *3*, 36
plotDMRsCircos, *3*, *29*, 37
plotDMRsCircos(), *31, 32*
plotDMRsManhattan, *3*, 40

readCustomMethylationBedData, 41
reduceBSseqAroundSeeds, 43

scoreDMRs, 44
simulateDMRs, 46
sortBetaFileByCoordinates, 49
sparsifySeeds, 51
stats::p.adjust(), *16*