# Package 'DAMEfinder'

November 11, 2026

**Type** Package

**Title** Finds DAMEs - Differential Allelicly MEthylated regions

**Version** 1.24.0

**Description** 'DAMEfinder' offers functionality for taking methtuple or bismark outputs to calculate ASM scores and compute DAMEs.
It also offers nice visualization of methyl-circle plots.

**License** MIT + file LICENSE

**LazyData** FALSE

**RoxygenNote** 7.2.2

**biocViews** DNAMethylation, DifferentialMethylation, Coverage

**Depends** R (>= 4.0)

**Imports** stats, Seqinfo, GenomicRanges, IRanges, S4Vectors, readr, SummarizedExperiment, GenomicAlignments, stringr, plyr, VariantAnnotation, parallel, ggplot2, Rsamtools, BiocGenerics, methods, limma, bumphunter, Biostrings, reshape2, cowplot, utils

**VignetteBuilder** knitr

**Suggests** BiocStyle, knitr, rmarkdown, testthat, rtracklayer, BSgenome.Hsapiens.UCSC.hg19

**BugReports** https://github.com/markrobinsonuzh/DAMEfinder/issues

**git_url** https://git.bioconductor.org/packages/DAMEfinder

**git_branch** RELEASE_3_23

**git_last_commit** 6f998f7

**git_last_commit_date** 2026-04-28

**Repository** Bioconductor 3.23

**Date/Publication** 2026-11-11

**Author** Stephany Orjuela [aut, cre] (ORCID: <https://orcid.org/0000-0002-1508-461X>), Dania Machlab [aut], Mark Robinson [aut]

**Maintainer** Stephany Orjuela <sorjuelal@gmail.com>

# Contents

calc_asm . . . 2
calc_derivedasm . . . 3
calc_logodds . . . 4
calc_score . . . 5
calc_weight . . . 5
DAMEfinder . . . 7
dame_track . . . 7
dame_track_mean . . . 8
empirical_pval . . . 9
extractbams_output . . . 10
extract_bams . . . 11
find_dames . . . 12
getMD . . . 14
get_tstats . . . 15
methyl_circle_plot . . . 16
methyl_circle_plotCpG . . . 18
methyl_MDS_plot . . . 19
modulus_sqrt . . . 20
readtuples_output . . . 20
read_tuples . . . 21
simes_pval . . . 22
splitReads . . . 22

**Index** **23**

---

`calc_asm` *Calculate ASM Score*

---

## Description

This function takes in a list of samples resulting from the read_tuples function and returns a SummarizedExperiment of Allele-Specific Methylation (ASM) scores, where each row is a tuple and each column is a sample.

## Usage

```
calc_asm(
  sampleList,
  beta = 0.5,
  a = 0.2,
  transform = modulus_sqrt,
  coverage = 5,
  verbose = TRUE
)
```

### Arguments

| | |
|---|---|
| sampleList | List of samples returned from read_tuples |
| beta | The beta parameter used to calculate the weight in the ASM score. link{calc_weight} uses this parameter to penalize fully methylated or unmethylated tuples. Default = 0.5. |
| a | The distance from 0.5 allowed, where 0.5 is a perfect MM:UU balance for a tuple. In the default mode this value is set to 0.2, and we account for the instances where the balance is between 0.3 and 0.7. |
| transform | Transform the calculated tuple ASM scores. We use the modulus square root function which outputs the square root, while preserving the original sign. |
| coverage | Remove tuples with total reads below coverage. Default = 5. |
| verbose | If the function should be verbose. Default = TRUE. |

### Details

Calculates ASM score for a list of samples in the output format of the result of read_tuples This functions uses the following other functions: process, calcScore, calcWeight.

### Value

A SummarizedExperiment of ASM scores where the rows are all the tuples and the columns the sample names.

### Examples

```
DATA_PATH_DIR <- system.file('extdata', '.', package = 'DAMEfinder')
get_data_path <- function(file_name) file.path(DATA_PATH_DIR, file_name)

tuple_files <- list.files(DATA_PATH_DIR, '.tsv.gz')
tuple_files <- get_data_path(tuple_files)
ASM <- read_tuples(tuple_files, c('CRC1', 'NORM1'))
ASMscore <- calc_asm(ASM)
```

---

## calc_derivedasm *Calculate SNP-based ASM*

---

### Description

Combines all the GRangeslist generated in extract_bams into a RangedSummarizedExperiment object, and calculates SNP-based allele-specific methylation.

### Usage

```
calc_derivedasm(sampleList, cores = 1, verbose = TRUE)
```

### Arguments

| | |
|---|---|
| sampleList | List of samples returned from extract_bams. |
| cores | Number of cores to thread. |
| verbose | If the function should be verbose. |

### Value

RangedSummarizedExperiment containing in assays:

- der.ASM: matrix with SNP-based ASM
- snp.table: Matrix with SNP associated to the CpG site.
- ref.cov: Coverage of the 'reference' allele.
- alt.cov: Coevarage of the 'alternative' allele.
- ref.meth: Methylated reads from the 'reference' allele.
- alt.meth: Methylated reads from the 'alternative' allele.

### Examples

```
data(extractbams_output)
derASM <- calc_derivedasm(extractbams_output[c(1,2)], cores = 1,
   verbose = FALSE)
```

---

## calc_logodds *Calculate the log odds ratio*

---

### Description

This function calculates the log odds ratio for a CpG tuple: (MM*UU)/(UM*MU), where 'M' stands for methylated and 'U' for unmethylated. 'MM' reflects the count for instances the CpG pair is methylated at both positions. The higher the MM and UU counts for that CpG pair, the higher the log odds ratio.

### Usage

```
calc_logodds(s, eps = 1)
```

### Arguments

| | |
|---|---|
| s | A data frame that contains the MM,UU,UM, and MU counts for each CpG tuple for a particular sample. It is the resulting object of the read_tuples. |
| eps | Count added to each of the MM,UU,UM and MU counts to avoid dividing by zero for example. The default is set to 1. |

### Value

The same object is returned with an additional column for the log odds ratio.

## calc_score *Calculate score*

### Description

This function calculates the ASM score for every tuple in a given sample. The ASM score is a multiplication of the log odds ratio by a weight that reflects the extent of allele-specific methylation. This weight is obtained with the `calc_weight` function.

### Usage

```
calc_score(df, beta = 0.5, a = 0.2)
```

### Arguments

| | |
|---|---|
| `df` | data frame of a sample containing all information per tuple (MM,UU,UM and MU counts, as well as the log odds ratio per tuple) needed for the ASM score. |
| `beta` | parameter for the `calc_weight` function. It's the alpha and beta values for the Beta function. |
| `a` | parameter for the calc_weight function. The weight will be the probability that the MM/(MM+UU) ratio lies between 0.5-a and 0.5+a. |

### Details

This function returns an allele-specific methylation (ASM) score for every given tuple in a sample. The ASM score is a product of the log odds ratio and a weight reflecting a measure of allele-specificity using the MM and UU counts.

### Value

The same object with an additional column for the ASM score.

## calc_weight *Calculate Weight for ASM Score*

### Description

This function calculates a weight which reflects MM to UU balance, where M stands for methylated and U for unmethylated. Given the MM and UU counts for a particular tuple, the weight is obtained using the `link{pbeta}` function.

### Usage

```
calc_weight(MM, UU, beta = 0.5, a = 0.2)
```

## Arguments

| | |
|---|---|
| MM | The read counts for where pos1 and pos2 of the tuple were both methylated. |
| UU | The read counts for where pos1 and pos2 of the tuple were both unmethylated. |
| beta | parameter for the beta distribution. In B(alpha,beta), we set alpha=beta=0.5 by default. |
| a | parameter for how far from 0.5 we go as a measure of allele-specific methylation. The weight is the probability that the MM:(MM+UU) ratio is between 0.5-a and 0.5+a. The default is set to 0.2. |

## Details

For a given tuple with MM and UU counts, the weight that reflects allele-scpecificity is calculated as follows:

- Prior:

$$p(\theta|\alpha,\beta) \sim Beta(\alpha,\beta),$$

where $\theta = \frac{MM}{MM+UU}$ and $\alpha = \beta = 0.5$. $p(\theta|\alpha,\beta)$ represents our prior belief which is that tuples are either fully methylated or fully unmethylated, rather than allele-specifically methylated which is a much rarer event.

- Likelihood:

$$p(x|\alpha,\beta) \propto \theta^{MM}(1-\theta)^{UU},$$

where x is our observation (the MM and UU counts).

- Posterior:

$$p(\theta|x) \propto p(x|\theta) * p(\theta|\alpha,\beta)$$

$$p(\theta|x) \propto \theta^{MM-0.5}(1-\theta)^{UU-0.5},$$

where $\alpha = \beta = 0.5$. This posterior also follows a beta distribution $\sim Beta(\alpha' = MM + 0.5, \beta' = UU + 0.5)$

## Value

A number that reflects allele-specificity given MM and UU counts for a CpG pair. This is used as a weight that is multiplied by the log odds ratio to give the final ASM score of that tuple.

#calc_weight(MM=50, UU=50) #0.9999716

#calc_weight(MM=20, UU=60) #0.1646916

DAMEfinder *DAMEfinder: Method to detect allele-specific methylation (ASM), and differential ASM from Bisulfite sequencing data in R.*

## Description

The package allows the user to extract an ASM score in two ways: either from a bismark bam file(s) and VCF file(s), or from the output from methtuple. Either way the final output is a list of regions with diferential allele-specific methylated between groups of samples of interest. The package also provides functions to visualize ASM at the read level or the score level

## DAMEfinder functions

calc_asm extracts ASM for pairs of CpG sites from a methtuple file, calc_derivedasm extracts ASM at each CpG site linked to a SNP from the VCF file. Both functions generate a RangedSummarizedExperiment, which is the input for the main function find_dames, that generates a data.frame with regions exhibiting differential ASM between a number of samples.

## Author(s)

Stephany Orjuela <sorjuelal@gmail.com>

Dania Machlab

Mark D Robinson <mark.robinson@imls.uzh.ch>

dame_track *Plot score tracks*

## Description

Plot score tracks

## Usage

```
dame_track(
  dame,
  window = 0,
  positions = 0,
  derASM = NULL,
  ASM = NULL,
  colvec = NULL,
  plotSNP = FALSE
)
```

## Arguments

| | |
|---|---|
| dame | GRanges object containing a region of interest, or detected with find_dames |
| window | Number of CpG sites outside (up or down-stream) of the DAME should be plotted. Default = 0. |
| positions | Number of bp sites outside (up or down-stream) of the DAME should be plotted. Default = 0. |
| derASM | SummarizedExperiment object obtained from calc_derivedasm (Filtering should be done by the user) |
| ASM | SummarizedExperiment object obtained from calc_asm (Filtering should be done by the user) |
| colvec | Vector of colors (mainly useful for the SNP plot, because I add it with cowplot, so I don't export a ggplot, optional) |
| plotSNP | whether to add the SNP track, only if derASM is specified. Default = FALSE |

## Value

Plot

## Examples

```
library(GenomicRanges)
DAME <- GRanges(19, IRanges(306443,310272))
data('readtuples_output')
ASM <- calc_asm(readtuples_output)
SummarizedExperiment::colData(ASM)$group <- c(rep('CRC',3),rep('NORM',2))
SummarizedExperiment::colData(ASM)$samples <- colnames(ASM)
dame_track(dame = DAME,
           ASM = ASM)
```

---

# dame_track_mean *Plot means per group of score tracks.*

---

## Description

Plot means per group of score tracks.

## Usage

```
dame_track_mean(
  dame,
  window = 0,
  positions = 0,
  derASM = NULL,
  ASM = NULL,
  colvec = NULL
)
```

### Arguments

| | |
|---|---|
| dame | GRanges object containing a region of interest, or detected with find_dames |
| window | Number of CpG sites outside (up or down-stream) of the DAME should be plotted. Default = 0. |
| positions | Number of bp sites outside (up or down-stream) of the DAME should be plotted. Default = 0. |
| derASM | SummarizedExperiment object obtained from calc_derivedasm (Filtering should be done by the user) |
| ASM | SummarizedExperiment object obtained from calc_asm (Filtering should be done by the user) |
| colvec | Vector of colors (mainly useful for the SNP plot, because I add it with cowplot, so I don't export a ggplot, optional) |

### Value

Plot

### Examples

```
library(GenomicRanges)
DAME <- GRanges(19, IRanges(306443,310272))
data('readtuples_output')
ASM <- calc_asm(readtuples_output)
SummarizedExperiment::colData(ASM)$group <- c(rep('CRC',3),rep('NORM',2))
SummarizedExperiment::colData(ASM)$samples <- colnames(ASM)
dame_track_mean(dame = DAME,
                ASM = ASM)
```

---

## empirical_pval *Calculate empirical region-level p-value*

---

### Description

This function permutes the coefficient of interest and re-runs `get_tstats` and `regionFinder` for each permutation. Code for permutations copied from the `dmrseq` function from the package of the same name.

### Usage

```
empirical_pval(
  presa,
  design,
  rforiginal,
  coeff,
  cont,
```

```
  smooth,
  maxPerms = 10,
  Q,
  maxGap,
  method,
  ...
)
```

## Arguments

| | |
|---|---|
| presa | SExperiment output from calc_derivedasm or calc_asm. |
| design | design matrix. |
| rforiginal | data.frame of DAMEs calculated with original design. |
| coeff | Coefficient of interest to permute. |
| cont | same as in get_tstats. |
| smooth | Boolean. |
| maxPerms | Maximum possible permutations generated. Default = 10. |
| Q | Quantile for cuttof. |
| maxGap | Same as other functions in the package. |
| method | lmFit method. |
| ... | Passed to get_tstats and then to loessByCluster. |

## Value

Vector of empirical p-values.

---

# extractbams_output *extract_bams() output.*

---

## Description

4 Patients from a previous study (Parker et al, 2018.) with colorectal cancer were sequenced and the normal and cancerous tissue of each patient was obtained. The data includes a subset of chromosome 19.

## Usage

```
extractbams_output
```

### Format

A large list with 8 elements. Each element is a list of GRanges for each sample. Each GRanges in the list includes the location of the CpG sites contained in the reads for each SNP. The GRanges metadata table contains:

`cov.ref` Number of reads of "reference" allele in that SNP

`cov.alt` Number of reads of "alternative" allele in that SNP

`meth.ref` Number of methylated reads of "reference" allele in that SNP

`cov.ref` Number of methylated reads of "alternative" allele in that SNP

`snp` The SNP containing the reads

For further details, see https://www.ebi.ac.uk/arrayexpress/experiments/E-MTAB-6949/ sample names in in ArrayExpress do not necessarily match names given here!

---

## `extract_bams` *Detect allele-specific methylation from a bam file*

---

### Description

The function takes a bam (from bismark) and vcf file for each sample. For each SNP contained in the vcfile it calculates the proportion of methylated reads for each CpG site at each allele. At the end it returns (saves to working directory) a GRanges list, where each GRanges contains all the CpG sites overlapping the reads containing a specific SNP.

### Usage

```
extract_bams(
  bamFiles,
  vcfFiles,
  sampleNames,
  referenceFile,
  coverage = 4,
  cores = 1,
  verbose = TRUE
)
```

### Arguments

| | |
|---|---|
| `bamFiles` | List of bam files. |
| `vcfFiles` | List of vcf files. |
| `sampleNames` | Names of files in the list. |
| `referenceFile` | fasta file used to generate the bam files. Or `DNAStringSet` with DNA sequence. |
| `coverage` | Minimum number of reads covering a CpG site on each allele. Default = 2. |
| `cores` | Number of cores to use. See package parallel for description of core. Default = 1. |
| `verbose` | Default = TRUE |

## Value

A list of GRanges for each sample. Each list is saved in a separate .rds file.

## Examples

```
DATA_PATH_DIR <- system.file('extdata', '.', package = 'DAMEfinder')
get_data_path <- function(file_name) file.path(DATA_PATH_DIR, file_name)
bamFiles <- get_data_path('NORM1_chr19_trim.bam')
vcfFiles <- get_data_path('NORM1.chr19.trim.vcf')
sampleNames <- 'NORM1'

#referenceFile
suppressPackageStartupMessages({library(BSgenome.Hsapiens.UCSC.hg19)})
genome <- BSgenome.Hsapiens.UCSC.hg19
seqnames(genome) <- gsub("chr","",seqnames(genome))
dna <- DNAStringSet(genome[[19]], use.names = TRUE)
names(dna) <- 19

GRanges_list <- extract_bams(bamFiles, vcfFiles, sampleNames, dna)
```

---

`find_dames` *Find DAMEs*

---

## Description

This function finds Differential Allele-specific MEthylated regions (DAMEs). It uses the `regionFinder` function from `bumphunter`, and asigns p-values either empirically or using the Simes method.

## Usage

```
find_dames(
  sa,
  design,
  coef = 2,
  contrast = NULL,
  smooth = TRUE,
  Q = 0.5,
  pvalAssign = "simes",
  maxGap = 20,
  verbose = TRUE,
  maxPerms = 10,
  method = "ls",
  trend = FALSE,
  ...
)
```

## Arguments

| | |
|---|---|
| sa | A SummarizedExperiment containing ASM values where each row correspond to a tuple/site and a column to sample/replicate. |
| design | A design matrix created with model.matrix. |
| coef | Column in design specifying the parameter to estimate. Default = 2. |
| contrast | a contrast matrix, generated with makeContrasts. |
| smooth | Whether smoothing should be applied to the t-Statistics. Default = TRUE. |
| Q | The percentile set to get a cutoff value K. K is the value on the Qth quantile of the absolute values of the given (smoothed) t-statistics. Only necessary if pvalAssign = 'empirical'. Default = 0.5. |
| pvalAssign | Choose method to assign pvalues, either 'simes' (default) or 'empirical'. This second one performs maxPerms number of permutations to calculate null statistics, and runs regionFinder. |
| maxGap | Maximum gap between CpGs in a cluster (in bp). NOTE: Regions can be as small as 1 bp. Default = 20. |
| verbose | If the function should be verbose. Default = TRUE. |
| maxPerms | Maximum possible permutations generated. Only necessary if pvalAssign = 'empirical'. Default = 10. |
| method | The method to be used in limma's lmFit. The default is set to 'ls' but can also be set to 'robust', which is recommended on a real data set. |
| trend | Passed to eBayes. Should an intensity-trend be allowed for the prior variance? Default is that the prior variance is constant, e.g. FALSE. |
| ... | Arguments passed to get_tstats. |

## Details

The simes method has higher power to detect DAMEs, but the consistency in signal across a region is better controlled with the empirical method, since it uses regionFinder and getSegments to find regions with t-statistics above a cuttof (controled with parameter Q), whereas with the 'simes' option, we initially detects clusters of CpG sites/tuples, and then test if at least 1 differential site/tuple is present in the cluster.

We recommend trying out different maxGap and Q parameters, since the size and the effect-size of obtained DAMEs change with these parameters.

## Value

A data frame of detected DAMEs ordered by the p-value. Each row is a DAME and the following information is provided in the columns (some column names change depending on the pvalAssign choice):

- chr: on which chromosome the DAME is found.
- start: The start position of the DAME.
- end: The end position of the DAME.
- pvalSimes: p-value calculated with the Simes method.

- pvalEmp: Empirical p-value obtained from permuting covariate of interest.
- sumTstat: Sum of t-stats per segment/cluster.
- meanTstat: Mean of t-stats per segment/cluster.
- segmentL: Size of segmented cluster (from `getSegments`).
- clusterL: Size of original cluster (from `clusterMaker`).
- FDR: Adjusted p-value using the method of Benjamini, Hochberg. (from `p.adjust`).
- numup: Number of sites with ASM increase in cluster (only for Simes).
- numdown: Number of sites with ASM decrease in cluster (only for Simes).

## Examples

```
data(readtuples_output)
ASM <- calc_asm(readtuples_output)
grp <- factor(c(rep('CRC',3),rep('NORM',2)), levels = c('NORM', 'CRC'))
mod <- model.matrix(~grp)
dames <- find_dames(ASM, mod, verbose = FALSE)
```

---

`getMD` *MDtag parser*

---

## Description

Takes a GenomicAlignments object containing the MDtag, and transforms it into a vector of characters and numbers

## Usage

```
getMD(a)
```

## Arguments

| | |
|---|---|
| a | Vector of MDtags (single characters) |

## Value

A named list of vectors, each vector a parsed version of MDtag: - nucl.num: Numeric representation of MDtag. - MDtag: a split version of MDtag

get_tstats *Get t-Statistics*

## Description

This function calculates a moderated t-Statistic per site or tuple using limma's lmFit and eBayes functions. It then smoothes the obtained t-Statistics using bumphunter's smoother function.

## Usage

```
get_tstats(
  sa,
  design,
  contrast = NULL,
  method = "ls",
  trend = FALSE,
  smooth = FALSE,
  maxGap = 20,
  coef = 2,
  verbose = TRUE,
  filter = TRUE,
  ...
)
```

## Arguments

| | |
|---|---|
| sa | A SummarizedExperiment containing ASM values where each row and column correspond to a tuple/site and sample respectively. |
| design | a design matrix created with model.matrix. |
| contrast | a contrast matrix, generated with makeContrasts. |
| method | The method to be used in limma's lmFit. The default is set to 'ls' but can also be set to 'robust', which is recommended on a real data set. |
| trend | Passed to eBayes. Should an intensity-trend be allowed for the prior variance? Default is that the prior variance is constant, e.g. FALSE. |
| smooth | Whether smoothing should be applied to the t-Statistics. Default = FALSE. If TRUE, wherever smoothing is not possible, the un-smoothed t-stat is used instead. |
| maxGap | The maximum allowed gap between genomic positions for clustering of genomic regions to be used in smoothing. Default = 20. |
| coef | Column in model.matrix specifying the parameter to estimate. Default = 2. If contrast specified, column with contrast of interest. |
| verbose | Set verbose. Default = TRUE. |
| filter | Remove empty tstats. Default = TRUE. |
| ... | Arguments passed to loessByCluster. Only used if smooth = TRUE. |

## Details

The smoothing is done on genomic clusters consisting of CpGs that are close to each other. In the case of tuples, the midpoint of the two genomic positions in each tuple is used as the genomic position of that tuple, to perform the smoothing.The function takes a `RangedSummarizedExperiment` generated by `calc_derivedasm` or `calc_asm` containing ASM across samples, and the index of control and treatment samples.

## Value

A vector of t-Statistics within the `RangedSummarizedExperiment`.

## Examples

```
data(readtuples_output)
ASM <- calc_asm(readtuples_output)
grp <- factor(c(rep('CRC',3),rep('NORM',2)), levels = c('NORM', 'CRC'))
mod <- model.matrix(~grp)
tstats <- get_tstats(ASM, mod)
```

---

# methyl_circle_plot *Draw methylation circle plot*

---

## Description

Draws CpG site methylation status as points, in reads containing a specific SNP. Generates one plot per bam file.

## Usage

```
methyl_circle_plot(
  snp,
  vcfFile,
  bamFile,
  refFile,
  build = "hg19",
  dame = NULL,
  letterSize = 2.5,
  pointSize = 3,
  sampleName = "sample1",
  cpgsite = NULL,
  sampleReads = FALSE,
  numReads = 20
)
```

## Arguments

| | |
|---|---|
| snp | GRanges object containing SNP location. |
| vcfFile | vcf file. |
| bamFile | bismark bam file path. |
| refFile | fasta reference file path. Or DNAStringSet with DNA sequence. |
| build | genome build used. default = "hg19" |
| dame | (optional) GRanges object containing a region to plot. |
| letterSize | Size of alleles drawn in plot. Default = 2.5. |
| pointSize | Size of methylation circles. Default = 3. |
| sampleName | FIX?: this is to save the vcf file to not generate it every time you run the function. |
| cpgsite | (optional) GRanges object containing a single CpG site location of interest. |
| sampleReads | Whether a subset of reads should be plotted. Default = FALSE. |
| numReads | Number of reads to plot per allele, if sampleReads is TRUE. Default = 20 |

## Value

Plot

## Examples

```
DATA_PATH_DIR <- system.file('extdata', '.', package = 'DAMEfinder')

get_data_path <- function(file_name) file.path(DATA_PATH_DIR, file_name)
bam_files <- get_data_path('NORM1_chr19_trim.bam')
vcf_files <- get_data_path('NORM1.chr19.trim.vcf')
sample_names <- 'NORM1'
#reference_file
suppressPackageStartupMessages({library(BSgenome.Hsapiens.UCSC.hg19)})
genome <- BSgenome.Hsapiens.UCSC.hg19
seqnames(genome) <- gsub("chr","",seqnames(genome))
dna <- DNAStringSet(genome[[19]], use.names = TRUE)
names(dna) <- 19

snp <- GenomicRanges::GRanges(19, IRanges::IRanges(292082, width = 1))
methyl_circle_plot(snp = snp,
 vcfFile = vcf_files,
 bamFile = bam_files,
 refFile = dna,
 sampleName = sample_names)
```

`methyl_circle_plotCpG` *Draw methylation circle plot without SNP*

## Description

Draws CpG site methylation status as points, in reads containing a specific CpG site. Generates one plot per bam file.

## Usage

```
methyl_circle_plotCpG(
  cpgsite = cpgsite,
  bamFile = bamFile,
  pointSize = 3,
  refFile = refFile,
  dame = NULL,
  order = FALSE,
  sampleName = NULL,
  sampleReads = FALSE,
  numReads = 20
)
```

## Arguments

| | |
|---|---|
| `cpgsite` | GRanges object containing a single CpG site location of interest |
| `bamFile` | bismark bam file path |
| `pointSize` | Size of methylation circles. Default = 3. |
| `refFile` | fasta reference file path |
| `dame` | (optional) GRanges object containing a region to plot |
| `order` | Whether reads should be sorted by methylation status. Default= False. |
| `sampleName` | Plot title. |
| `sampleReads` | Whether a subset of reads should be plotted. Default = FALSE. |
| `numReads` | Number of reads to plot, if sampleReads is TRUE. Default = 20 |

## Value

Plot

## Examples

```
DATA_PATH_DIR <- system.file('extdata', '.', package = 'DAMEfinder')
get_data_path <- function(file_name) file.path(DATA_PATH_DIR, file_name)
bam_files <- get_data_path('NORM1_chr19_trim.bam')
sample_names <- 'NORM1'
#reference_file
```

```
suppressPackageStartupMessages({library(BSgenome.Hsapiens.UCSC.hg19)})
genome <- BSgenome.Hsapiens.UCSC.hg19
seqnames(genome) <- gsub("chr","",seqnames(genome))
dna <- DNAStringSet(genome[[19]], use.names = TRUE)
names(dna) <- 19

cpg <- GenomicRanges::GRanges(19, IRanges::IRanges(292082, width = 1))
methyl_circle_plotCpG(cpgsite = cpg,
 bamFile = bam_files,
 refFile = dna)
```

---

`methyl_MDS_plot` *Multidimensional scaling plot of distances between methylation proportions (beta values)*

---

## Description

Same as `plotMDS`, except for an arc-sine transformation of the methylation proportions.

## Usage

```
methyl_MDS_plot(x, group, top = 1000, coverage = 5, adj = 0.02, pointSize = 4)
```

## Arguments

| | |
|---|---|
| `x` | `RangedSummarizedExperiment`, output from `calc_derivedasm` or `calc_asm`. |
| `group` | Vector of group or any other labels, same length as number of samples. |
| `top` | Number of top CpG sites used to calculate pairwise distances. |
| `coverage` | Minimum number of reads covering a CpG site on each allele. Default = 5. |
| `adj` | Text adjustment in y-axis. Default = 0.2. |
| `pointSize` | Default = 4. |

## Value

Two-dimensional MDS plot.

## Examples

```
data(readtuples_output)
ASM <- calc_asm(readtuples_output)
grp <- factor(c(rep('CRC',3),rep('NORM',2)), levels = c('NORM', 'CRC'))
methyl_MDS_plot(ASM, grp)
```

## modulus_sqrt *Get Modulus Square Root*

### Description

Function to calculate signed square root (aka modulus square root).

### Usage

```
modulus_sqrt(values)
```

### Arguments

| | |
|---|---|
| values | Vector or matrix of ASM scores where each column is a sample. These values are transformed with a square root transformation that (doesn't) preserve the sign. |

### Value

Vector or matrix of transformed scores.

## readtuples_output *read_tuples() output.*

### Description

3 Patients from a previous study (Parker et al, 2018.) with colorectal cancer were sequenced and the normal and cancerous tissue of each patient was obtained. The data includes a subset of chromosome 19. Here one normal sample is not included.

### Usage

```
readtuples_output
```

### Format

A large list with 5 elements. Each element is a `tibble` with the coordinates of the pairs of CpG sites (tuples). Rest of the tibble contains:

`MM` Number of reads with both CpG sites methylated

`MU` Number of reads with first CpG site methylated

`UM` Number of reads with second CpG site methylated

`UU` Number of reads with both CpG sites unmethylated

`cov` Coverage, total reads at tuple

`inter_dist` Distance in bp between CpG sites

For further details, see https://www.ebi.ac.uk/arrayexpress/experiments/E-MTAB-6949/ sample names in in ArrayExpress do not necessarily match names given here!

read_tuples *Read in list of methtuple files*

## Description

This function reads in a list of files obtained from the methtuple tool. It filters out tuples based on the set minimum coverage (min_cov) and the maximum allowed distance (maxGap) between two genomic positions in a tuple.

## Usage

```
read_tuples(files, sampleNames, minCoverage = 2, maxGap = 20, verbose = TRUE)
```

## Arguments

| | |
|---|---|
| files | List of methtuple files. |
| sampleNames | Names of files in the list. |
| minCoverage | The minimum coverage per tuple. Tuples with a coverage < minCoverage are filtered out. Default = 2. |
| maxGap | The maximum allowed distance between two positions in a tuple. Only distances that are <= maxGap are kept. Default = 150 base pairs. |
| verbose | If the function should be verbose. |

## Value

A list of data frames, where each data frame corresponds to one file.

## Examples

```
DATA_PATH_DIR <- system.file('extdata', '.', package = 'DAMEfinder')
get_data_path <- function(file_name) file.path(DATA_PATH_DIR, file_name)

tuple_files <- list.files(DATA_PATH_DIR, '.tsv.gz')
tuple_files <- get_data_path(tuple_files)
ASM <- read_tuples(tuple_files, c('CRC1', 'NORM1'))
```

## simes_pval *Calculate region-level p-value*

### Description

This function uses the Simes method to calculate a regional-level p-value based on the single-eBayes p-values. It highly depends on the choice of `maxGap` in `find_dames`.

### Usage

```
simes_pval(sat, smtstat, midpt)
```

### Arguments

| | |
|---|---|
| `sat` | Output from `get_tstats`. |
| `smtstat` | (Smoothed) tstat vector from `get_tstats`. |
| `midpt` | Coordinate vector for each CpG site/tuple. |

### Details

When used as a FDR-control method, for positively correlated P-values, Simes method is even closer to the nominal alpha level than the Bonferroni-Holm method.

### Value

Vector of summarized pvals

## splitReads *Divide read names by allele*

### Description

Takes a GenomicAlignments object and returns a list of read names dividied by allele.

### Usage

```
splitReads(alns, v, snp)
```

### Arguments

| | |
|---|---|
| `alns` | GenomicAlignments object. |
| `v` | Nucleotide of reference (or alternative) allele. |
| `snp` | GRanges object containing SNP location. |

### Value

A named list of vectors, each vector containing read names for each allele.

# Index


∗ **datasets**
    extractbams_output, 10
    readtuples_output, 20
∗ **internal**
    empirical_pval, 9
    getMD, 14
    simes_pval, 22
    splitReads, 22

calc_asm, 2, *16*
calc_derivedasm, 3, *16*
calc_logodds, 4
calc_score, 5
calc_weight, *5*, 5
clusterMaker, *14*

dame_track, 7
dame_track_mean, 8
DAMEfinder, 7

eBayes, *13*, *15*
empirical_pval, 9
extract_bams, *3, 4*, 11
extractbams_output, 10

find_dames, 12

get_tstats, *9*, *13*, 15, *22*
getMD, 14
getSegments, *14*

lmFit, *13*, *15*
loessByCluster, *15*

makeContrasts, *13*, *15*
methyl_circle_plot, 16
methyl_circle_plotCpG, 18
methyl_MDS_plot, 19
model.matrix, *13*, *15*
modulus_sqrt, 20

p.adjust, *14*
plotMDS, *19*

RangedSummarizedExperiment, *3*
read_tuples, *3*, 21
readtuples_output, 20
regionFinder, *9*, *12*

simes_pval, 22
smoother, *15*
splitReads, 22