Package ‘ramr’

September 24, 2026

Title Detection of Rare Aberrantly Methylated Regions in Array and NGS Data

Version 1.20.2

Description ramr is an R package for detection of epimutations (i.e., infrequent aberrant DNA methylation events) in large data sets obtained by methylation profiling using array or high-throughput methylation sequencing. In addition, package provides functions to visualize found aberrantly methylated regions (AMRs), to generate sets of all possible regions to be used as reference sets for enrichment analysis, and to generate biologically relevant test data sets for performance evaluation of AMR/DMR search algorithms.

SystemRequirements C++20, GNU make

NeedsCompilation yes

Depends R (>= 4.1)

Imports methods, data.table, Seqinfo, GenomicRanges, IRanges, BiocGenerics, S4Vectors, Rcpp

LinkingTo Rcpp

Suggests RUnit, knitr, rmarkdown, ggplot2, gridExtra, annotatr, LOLA, org.Hs.eg.db, TxDb.Hsapiens.UCSC.hg19.knownGene, parallel, doParallel, foreach, doRNG, matrixStats, EnvStats, ExtDist, gamlss, gamlss.dist

License Artistic-2.0

URL https://github.com/BBCG/ramr

BugReports https://github.com/BBCG/ramr/issues

Encoding UTF-8

biocViews DNAMethylation, DifferentialMethylation, Epigenetics, MethylationArray, MethylSeq

VignetteBuilder knitr

Config/roxygen2/version 8.1.0

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

git_branch RELEASE_3_23

git_last_commit 554937a

git_last_commit_date 2026-09-16

Repository Bioconductor 3.23

Date/Publication 2026-09-23

Author Oleksii Nikolaienko [aut, cre] (ORCID: <https://orcid.org/0000-0002-5910-4934>)

Maintainer Oleksii Nikolaienko <oleksii.nikolaienko@gmail.com>

Contents

getAMR 2
getAMR.obsolete 8
getUniverse 10
plotAMR 11
ramr.data 13
simulateAMR 14
simulateData 16
simulateData.obsolete 19

Index **22**

`getAMR` *Search for aberrantly methylated regions*

Description

'getAMR' returns a 'GRanges' object with aberrantly methylated regions (AMRs / epimutations) for all samples in a data set.

Usage

```
getAMR(
  data.ranges,
  data.samples = NULL,
  data.coverage = NULL,
  transform = c("identity", "linear"),
  exclude.range = NULL,
  compute = c("IQR", "beta+binom"),
  compute.estimate = c("mom", "amle", "nmle"),
  compute.weights = c("equal", "logInvDist", "sqrtInvDist", "invDist"),
  combine = c("threshold", "comb-p"),
  combine.threshold = ifelse(compute == "IQR", 5, 0.001),
  combine.window = 300,
  combine.min.cpgs = 7,
```

```
  combine.min.width = 1,
  combine.ignore.strand = FALSE,
  ncores = NULL,
  verbose = TRUE
)
```

Arguments

`data.ranges` A ‘GRanges‘ object with genomic locations and corresponding beta values included as metadata.

`data.samples` A character vector with sample names (e.g., a subset of metadata column names). If ‘NULL‘ (the default), then all samples (metadata columns) are included in the analysis.

`data.coverage` An optional ‘data.frame‘ object with coverage data. If provided, must be an all-integer ‘data.frame‘, with the same dimensions and column names as the ‘data.ranges‘ metadata columns.

`transform` A character scalar specifying if beta values should be used as supplied ("identity", the default) or linearly transformed ("linear") to force all $\{0;1\}$ endpoint (extreme) values within open $(0,1)$ interval using the following formula:

$$x' = \frac{x(N-1)+0.5}{N}$$

where x is the beta value before transformation, and N is the number of samples. Such transformation is recommended only if beta values contain $\{0;1\}$ values **and** coverage data is NOT available. See doi: 10.1037/1082-989X.11.1.54 and Dealing with 0,1 values in a beta regression for more details.

`exclude.range` A numeric vector of length two. Unless ‘NULL‘ (the default), all ‘data.ranges‘ genomic locations with their median methylation beta value within the ‘exclude.range‘ interval are filtered out.

`compute` A character scalar: when "IQR" (the default), AMR search based on interquantile range is performed. When "beta+binom" - search is based on fitting optionally weighted beta distribution (with the possibility of estimating binomial probability of $\{0;1\}$ endpoint values). See Details section for explanation.

`compute.estimate` A character scalar for the method of parameter estimation of beta distribution. The default ("mom") stands for the method of moments based on the unbiased estimator of variance and includes $\{0;1\}$ endpoints in calculation of moments (mean, unbiased variance). Other options are "amle" (approximation of maximum likelihood estimation) and "nmle" (numeric maximum likelihood estimation) - both ignore $\{0;1\}$ endpoints in calculations. See Details section for explanation.

`compute.weights` A character scalar for the weights assigned to individual observations (beta values) and used to compute weighted means and variance as described in "Compute" section below. Four available weighing schemes result in different sensitivity of outlier detection and rate of false positive (FP) findings: "equal" (the default) is the least sensitive and gives least number of FPs, while "invDist" is

the most sensitive but may result in a very high number of FPs especially when ‘combine.threshold‘ is too high (1e-3 or higher). "logInvDist" is recommended when one desires a balance between relatively low type I error rate and higher detection sensitivity for both unique and non-unique AMRs.

If "equal", all weights are equal to 1 ($w_i = 1$). Otherwise, weights of observations inversely depend on their distance from the median (thus emphasizing outliers) and are calculated using the following formulas:

$$\text{"logInvDist": } w_i = \log \frac{1}{|M - x_i| + \epsilon}$$

$$\text{"sqrtInvDist": } w_i = \sqrt{\frac{1}{|M - x_i| + \epsilon}}$$

$$\text{"invDist": } w_i = \frac{1}{|M - x_i| + \epsilon}$$

where x_i is a beta value for i-th sample, M is a median of all beta values at this genomic location, and ϵ is a very small number (= FLT_EPSILON $\approx$ 1.192093e-07).

`combine` A character scalar for the method used to combine individual outlier genomic positions into genomic intervals (ranges). When "threshold" (the default) is used, simple thresholding is applied. More details are given in the "Combine" subsection below.

`combine.threshold` A numeric scalar setting the threshold for an outlier value. When ‘compute=="IQR"‘, methylation beta values differing from the median value by at least ‘combine.threshold‘ interquartile ranges are considered to be outliers (the default: 5). When ‘compute=="beta+binom"‘, all probability values not higher than ‘combine.threshold‘ are considered to be outliers (the default: 0.001).

`combine.window` A positive integer. All significant (survived the filtering stage) ‘data.ranges‘ genomic locations within this distance will be merged to create AMRs (the default: 300).

`combine.min.cpgs` A single integer >= 1. All AMRs containing less than ‘combine.min.cpgs‘ significant genomic locations are filtered out (the default: 7).

`combine.min.width` A single integer >= 1 (the default). Only AMRs with the width of at least ‘combine.min.width‘ are returned.

`combine.ignore.strand` A boolean scalar to ignore strand information in the input ‘data.ranges‘ and when outlier genomic positions are combined into genomic intervals (the default: FALSE).

`ncores` A single integer >= 1. Number of OpenMP threads for parallel computation (the default: half of the available cores). Results of parallel processing are fully reproducible.

`verbose` Boolean to report progress and timings (default: TRUE).

Details

In the provided data set, ‘getAMR‘ finds stretches of outlier beta values to identify rare long-range methylation aberrations (epimutations) in one or several samples. As a rule, methods for differential methylation analysis rely on between-group comparisons — ‘getAMR‘ performs this comparison within-group, which is not only faster, but also more sensitive. The logic of computations is described below.

Compute: This section describes computations that are performed in order to identify individual outlier beta values — but **before values are deemed as outliers**. At the moment, the two supported methods are:

"IQR" When ‘compute=="IQR"‘, for every genomic location (CpG) in ‘data.ranges‘ the IQR-normalized deviation from the median value is calculated using the following formula:

$$xIQR_i = \frac{x_i - M}{IQR}$$

where x_i is a beta value for i-th sample, M and IQR are a median and an interquartile range of all beta values at this genomic location, respectively.

"beta+binom" When ‘compute=="beta+binom"‘, for every genomic location (CpG), ‘getAMR‘ will estimate the probability of each beta value to occur. For all beta values inside the open $(0, 1)$ interval, beta distribution is used. For all $\{0; 1\}$ endpoint (extreme) values where beta distribution is not defined, binomial probability is calculated using coverage data (supplied using ‘data.coverage‘ parameter).
Alpha α and beta β parameters of beta distribution can be estimated using one of the following methods:

1. Method of moments (‘compute.estimate="mom"‘) based on (both optionally weighted) mean and unbiased variance

$$\text{sample mean} = \bar{x} = \frac{\sum\limits_{i=1}^{n} w_i x_i}{\sum\limits_{i=1}^{n} w_i}$$

$$\text{unbiased variance} = \bar{v} = \frac{\sum\limits_{i=1}^{N} w_i (x_i - \bar{x})^2}{\sum_{i=1}^{N} w_i - (\sum_{i=1}^{N} w_i^2 / \sum_{i=1}^{N} w_i)}$$

where x_i and w_i are a beta value and its reliability weight for i-th sample.

$$\hat{\alpha} = \bar{x} \left(\frac{\bar{x}(1 - \bar{x})}{\bar{v}} - 1 \right)$$

$$\hat{\beta} = (1 - \bar{x}) \left(\frac{\bar{x}(1 - \bar{x})}{\bar{v}} - 1 \right)$$

Note: all beta values from the closed $[0, 1]$ interval are used for calculation of arithmetic mean value. For more details on used formulas, visit Weighted arithmetic mean, Weighted variance with reliability weights, and Method of moments for beta distribution

2. Approximate maximum likelihood (‘compute.estimate="amle"‘) based on (both optionally weighted) geometric means of x and $(1-x)$

$$\hat{G}_x = \left(\prod_{i=1}^{n} x_i^{w_i}\right)^{1/\sum_{i=1}^{n} w_i}$$

$$\hat{G}_{(1-x)} = \left(\prod_{i=1}^{n} (1-x_i)^{w_i}\right)^{1/\sum_{i=1}^{n} w_i}$$

where x_i and w_i are a beta value and its reliability weight for i-th sample.

$$\hat{\alpha} \approx \tfrac{1}{2} + \frac{\hat{G}_x}{2(1-\hat{G}_x-\hat{G}_{(1-x)})}$$

$$\hat{\beta} \approx \tfrac{1}{2} + \frac{\hat{G}_{(1-x)}}{2(1-\hat{G}_x-\hat{G}_{(1-x)})}$$

Note: only beta values from the open $(0,1)$ interval are used for calculation of geometric means. For more details, visit Weighted geometric mean, and Maximum likelihood for beta distribution

3. And a numerical maximum likelihood (‘compute.estimate="nmle"‘) which is yet to be implemented.

After estimating parameters of beta distribution, probabilities of observed beta values from the open $(0,1)$ interval are computed using regularized incomplete beta function $I_x(\alpha,\beta)$. For more details, visit Cumulative distribution function for beta distribution.
Probabilities of $\{0;1\}$ endpoint values are computed using mean value (either arithmetic for ‘compute.estimate="mom"‘ or geometric for ‘compute.estimate="*mle"‘) using the following formulas:

$${p^0}_i = (1-\bar{x})^k$$

$${p^1}_i = \bar{x}^k$$

where ${p^0}_i$ and ${p^1}_i$ are probabilities of observing 0 or 1 for i-th sample, respectively, $\bar{x}$ is a mean of all beta values for this genomic position, and k is a sequencing coverage of this genomic position for i-th sample.

Combine: This section describes how individual beta values are deemed as outliers and how these outliers are combined into extended genomic regions (aberrantly methylated regions, AMRs, or epimutations). The only method supported at the moment is thresholding as described below.

"threshold" This method applies a fixed threshold (specified using ‘combine.threshold‘ parameter) to all ‘xIQR‘ or probability values computed at the previous step. All observed beta values that are passing this threshold (above or equal to the threshold for ‘xIQR‘; below or equal in case of probability values) are considered to be outliers and therefore retained.
Next, for all outlier beta values per sample, corresponding genomic positions are merged into genomic intervals using the window of ‘combine.window‘ and keeping the strand information unless ‘combine.ignore.strand‘ is TRUE.

"comb-p" This method of combining outliers into genomic ranges is yet to be implemented.

Resulting genomic intervals are then filtered: only the regions containing at least ‘combine.min.cpgs‘ outliers and which are at leas as wide as ‘combine.min.width‘ are reported back.

As a final step, the following average values are computed for every aberrantly methylated region:

1. arithmetic mean of distances of all outlier beta values to a median beta value (‘dbeta‘)
2. if ‘compute=="IQR"‘, arithmetic mean of xIQR values of all outlier genomic position (‘xiqr‘) OR
3. if ‘compute=="beta+binom"‘, geometric mean of probability values of all outlier genomic position (‘pval‘)

Value

The output is a ‘GRanges‘ object that contains all the aberrantly methylated regions (AMRs / epimutations) for all ‘data.samples‘ samples in ‘data.ranges‘ object. The following metadata columns may be present:

- ‘revmap‘ – integer list of significant CpGs (‘data.ranges‘ genomic locations) that are included in this AMR region
- ‘ncpg‘ – number of significant CpGs within this AMR region
- ‘sample‘ – contains an identifier of a sample to which corresponding AMR belongs
- ‘dbeta‘ – average deviation of beta values for significant CpGs from their corresponding median values
- ‘pval‘ – geometric mean of p-values for significant CpGs
- ‘xiqr‘ – average IQR-normalised deviation of beta values for significant CpGs from their corresponding median values

Note

NA values within metadata columns of ‘data.ranges‘ are silently dropped in all computations.

See Also

plotAMR for plotting AMRs, getUniverse for info on enrichment analysis, simulateAMR and simulateData for the generation of simulated test data sets, and ‘ramr‘ vignettes for the description of usage and sample data.

Examples

```
data(ramr)
getAMR(data.ranges=ramr.data, data.samples=ramr.samples,
       compute="beta+binom", compute.estimate="amle",
       combine.min.cpgs=5, combine.window=1000, combine.threshold=1e-3)
```

getAMR.obsolete *[OBSOLETE] Search for aberrantly methylated regions*

Description

This function is fully functional but obsolete. It will remain a part of the package for consistency, as it was used in ‘ramr‘ publication (doi:10.1093/bioinformatics/btab586). Please use faster and more capable getAMR instead.

‘getAMR.obsolete‘ returns a ‘GRanges‘ object with all the aberrantly methylated regions (AMRs) for all samples in a data set.

Usage

```
getAMR.obsolete(
  data.ranges,
  data.samples = NULL,
  ramr.method = c("IQR", "beta", "wbeta", "beinf"),
  iqr.cutoff = 5,
  pval.cutoff = 0.05,
  qval.cutoff = NULL,
  merge.window = 300,
  min.cpgs = 7,
  min.width = 1,
  exclude.range = NULL,
  cores = max(1, parallel::detectCores() - 1),
  verbose = TRUE,
  ...
)
```

Arguments

data.ranges	A ‘GRanges‘ object with genomic locations and corresponding beta values included as metadata.
data.samples	A character vector with sample names (a subset of metadata column names). If ‘NULL‘ (the default), then all samples (metadata columns) are included in the analysis.
ramr.method	A character scalar: when ramr.method is "IQR" (the default), the filtering based on interquantile range is used (‘iqr.cutoff‘ value is then used as a threshold). When "beta", "wbeta" or "beinf" - filtering based on fitting non-weighted (‘EnvStats::ebeta‘), weighted (‘ExtDist::eBeta‘) or zero-and-one inflated (‘gamlss.dist::BEINF‘) beta distributions, respectively, is used, and ‘pval.cutoff‘ or ‘qval.cutoff‘ (if not ‘NULL‘) is used as a threshold. For "wbeta", weights directly correlate with bin contents (number of values per bin) and inversly - with the distances from the median value, thus narrowing the estimated distribution and emphasizing outliers.

`iqr.cutoff`	A single integer >= 1. Methylation beta values differing from the median value by more than ‘iqr.cutoff‘ interquartile ranges are considered to be significant (the default: 5).
`pval.cutoff`	A numeric scalar (the default: 5e-2). Bonferroni correction of ‘pval.cutoff‘ by the length of the ‘data.samples‘ object is used to calculate ‘qval.cutoff‘ if the latter is ‘NULL‘.
`qval.cutoff`	A numeric scalar. Used as a threshold for filtering based on fitting non-weighted or weighted beta distributions: all p-values lower than ‘qval.cutoff‘ are considered to be significant. If ‘NULL‘ (the default), it is calculated using ‘pval.cutoff‘
`merge.window`	A positive integer. All significant (survived the filtering stage) ‘data.ranges‘ genomic locations within this distance will be merged to create AMRs (the default: 300).
`min.cpgs`	A single integer >= 1. All AMRs containing less than ‘min.cpgs‘ significant genomic locations are filtered out (the default: 7).
`min.width`	A single integer >= 1 (the default). Only AMRs with the width of at least ‘min.width‘ are returned.
`exclude.range`	A numeric vector of length two. If not ‘NULL‘ (the default), all ‘data.ranges‘ genomic locations with their median methylation beta value within the ‘exclude.range‘ interval are filtered out.
`cores`	A single integer >= 1. Number of processes for parallel computation (the default: all but one cores). Results of parallel processing are fully reproducible when the same seed is used (thanks to doRNG).
`verbose`	boolean to report progress and timings (default: TRUE).
`...`	Further arguments to be passed to ‘EnvStats::ebeta‘ or ‘ExtDist::eBeta‘ functions.

Details

In the provided data set, ‘getAMR.obsolete‘ compares methylation beta values of each sample with other samples to identify rare long-range methylation aberrations (epimutations). For ‘ramr.method=="IQR"‘: for every genomic location (CpG) in ‘data.ranges‘ the IQR-normalized deviation from the median value is calculated, and all CpGs with such normalized deviation not smaller than the ‘iqr.cutoff‘ are retained. For ‘ramr.method distribution are estimated by means of ‘EnvStats::ebeta‘ (beta distribution), ‘ExtDist::eBeta‘ (weighted beta destribution), or ‘gamlss.dist::BEINF‘ (zero and one inflated beta distribution) functions, respectively. These parameters are then used to calculate the probability values, followed by the filtering when all CpGs with p-values not greater than ‘qval.cutoff‘ are retained. Another filtering is then performed to exclude all CpGs within ‘exclude.range‘. Next, the retained (significant) CpGs are merged within the window of ‘merge.window‘, and final filtering is applied to AMR genomic ranges (by ‘min.cpgs‘ and ‘min.width‘).

Value

The output is a ‘GRanges‘ object that contains all the aberrantly methylated regions (AMRs) for all ‘data.samples‘ samples in ‘data.ranges‘ object. The following metadata columns may be present:

- ‘revmap‘ – integer list of significant CpGs (‘data.ranges‘ genomic locations) that are included in this AMR region

- ‘ncpg‘ – number of significant CpGs within this AMR region
- ‘sample‘ – contains an identifier of a sample to which corresponding AMR belongs
- ‘dbeta‘ – average deviation of beta values for significant CpGs from their corresponding median values
- ‘pval‘ – geometric mean of p-values for significant CpGs
- ‘xiqr‘ – average IQR-normalised deviation of beta values for significant CpGs from their corresponding median values

See Also

plotAMR for plotting AMRs, getUniverse for info on enrichment analysis, simulateAMR and simulateData for the generation of simulated test data sets, and ‘ramr‘ vignettes for the description of usage and sample data.

Examples

```
data(ramr)
getAMR.obsolete(ramr.data, ramr.samples, ramr.method="beta",
                min.cpgs=5, merge.window=1000, qval.cutoff=1e-3, cores=2)
```

---

getUniverse — *Merges, filters and outputs all genomic regions of a given ‘GRanges‘ object*

Description

‘getUniverse‘ returns a ‘GRanges‘ object with all the genomic regions in a data set, that can be used for AMR enrichment analysis

Usage

```
getUniverse(data.ranges, merge.window = 300, min.cpgs = 7, min.width = 1)
```

Arguments

data.ranges	A ‘GRanges‘ object with genomic locations and corresponding beta values included as metadata.
merge.window	A single integer >= 1. All ‘data.ranges‘ genomic locations within this distance will be merged (the default: 300).
min.cpgs	A single integer >= 1. All genomic regions containing less than ‘min.cpgs‘ genomic locations are filtered out (the default: 7).
min.width	A single integer >= 1 (the default). Only regions with the width of at least ‘min.width‘ are returned.

Details

In the provided data set ‘getUniverse‘ merges and outputs all the genomic regions that satisfy filtering criteria, thus creating a ‘GRanges‘ object to be used as a reference set of genomic regions for AMR enrichment analysis.

Value

The output is a ‘GRanges‘ object that contain all the genomic regions in ‘data.ranges‘ object (in other words, all potential AMRs).

See Also

getAMR for identification of AMRs, plotAMR for plotting AMRs, simulateAMR and simulateData for the generation of simulated test data sets, and ‘ramr‘ vignettes for the description of usage and sample data.

Examples

```
data(ramr)
universe <- getUniverse(ramr.data, min.cpgs=5, merge.window=1000)

# identify AMRs
amrs <- getAMR(
  data.ranges=ramr.data, compute="beta+binom",
  combine.min.cpgs=5, combine.window=1000, combine.threshold=1e-3
)

# AMR enrichment analysis using LOLA
library(LOLA)
# download LOLA region databases from http://databio.org/regiondb
hg19.extdb.file <- system.file("LOLAExt", "hg19", package="LOLA")
if (file.exists(hg19.extdb.file)) {
  hg19.extdb  <- loadRegionDB(hg19.extdb.file)
  runLOLA(amrs, universe, hg19.extdb, cores=1, redefineUserSets=TRUE)
}
```

---

plotAMR *Plot aberrantly methylated regions*

Description

‘plotAMR‘ uses ‘ggplot2‘ to visualize aberrantly methylated regions (AMRs) at particular genomic locations.

Usage

```
plotAMR(
  data.ranges,
  amr.ranges,
  data.samples = NULL,
  window = 300,
  ignore.strand = FALSE,
  highlight = NULL,
  title = NULL,
  labs = c("genomic position", "beta value"),
  transform = c("identity", "log1p", "log10"),
  limits = NULL,
  breaks = NULL,
  verbose = TRUE
)
```

Arguments

data.ranges	A ‘GRanges‘ object with genomic locations and corresponding beta values included as metadata.
amr.ranges	An output of ‘getAMR‘ - a ‘GRanges‘ object that contain aberrantly methylated regions (AMRs).
data.samples	A character vector with sample names (a subset of metadata column names) to be included in the plot. If ‘NULL‘ (the default), then all samples (metadata columns) are included.
window	An optional integer constant to expand genomic ranges of the ‘amr.ranges‘ object (the default: 300).
ignore.strand	Boolean to ignore strand of AMR region. Default: FALSE.
highlight	An optional list of samples to highlight. If NULL (the default), will contain sample IDs from the ‘sample‘ metadata column of ‘amr.ranges‘ object.
title	An optional title for the plot. If NULL (the default), plot title is set to a genomic location of particular AMR.
labs	Optional axis labels for the plot. Default: c("genomic position", "beta value").
transform	Optional transformation of y-axis. Default: "identity" (no transformation).
limits	Optional limits of y-axis. When default (NULL), limits are c(NA,1) for ‘transform=="log10"‘ and c(0,1) otherwise.
breaks	Optional breaks of y-axis. When default (NULL), breaks are ‘10**(seq(from=-5, to=0, length.out=6))‘ for ‘transform=="log10"‘ and ‘seq(from=0, to=1, length.out=6)‘ otherwise.
verbose	Boolean to report progress and timings (default: TRUE).

Details

For every non-overlapping genomic location from ‘amr.ranges‘ object, ‘plotAMR‘ plots and outputs a line graph of methylation beta values taken from ‘data.ranges‘ for all samples from ‘data.samples‘. Samples bearing significantly different methylation profiles (’sample’ column of ‘amr.ranges‘ object) are highlighted.

Value

The output is a list of ‘ggplot‘ objects.

See Also

getAMR for identification of AMRs, getUniverse for info on enrichment analysis, simulateAMR and simulateData for the generation of simulated test data sets, and ‘ramr‘ vignettes for the description of usage and sample data.

Examples

```
data(ramr)
plotAMR(data.ranges=ramr.data, amr.ranges=ramr.tp.unique[1])
library(gridExtra)
do.call("grid.arrange",
        c(plotAMR(data.ranges=ramr.data, amr.ranges=ramr.tp.nonunique), ncol=2))
```

---

ramr.data — *Simulated Illumina HumanMethylation 450k data set with 3000 CpGs and 100 samples*

Description

Data was simulated using GSE51032 data set as described in the reference. Current data set ("ramr.data") contains beta values for 10000 CpGs and 100 samples ("ramr.samples"), and carries 6 unique ("ramr.tp.unique") and 15 non-unique ("ramr.tp.nonunique") true positive AMRs containing at least 10 CpGs with their beta values increased/decreased by 0.5.

Usage

```
data(ramr)
```

Format

Objects of class "GRanges" ("ramr.data, ramr.tp.unique,ramr.tp.nonunique") and "character" ("ramr.samples").

References

Nikolaienko et al., 2020 (bioRxiv)

Examples

```
data(ramr)
amrs <- getAMR(
  data.ranges=ramr.data, compute="IQR",
  combine.min.cpgs=5, combine.window=1000, combine.threshold=5
)
plotAMR(data.ranges=ramr.data, amr.ranges=amrs[1])
plotAMR(data.ranges=ramr.data, amr.ranges=ramr.tp.nonunique[4],
        highlight=c("sample7","sample8","sample9"))
```

`simulateAMR` *Simulate a set of aberrantly methylated regions*

Description

‘simulateAMR‘ returns a ‘GRanges‘ object containing a set of randomly selected aberrantly methylated regions (AMRs) to be used as an input for the ‘simulateData‘ method.

Usage

```
simulateAMR(
  template.ranges,
  nsamples,
  exclude.ranges = NULL,
  regions.per.sample = 1,
  samples.per.region = 1,
  sample.names = NULL,
  merge.window = 300,
  min.cpgs = 7,
  max.cpgs = Inf,
  min.width = 1,
  dbeta = 0.25
)
```

Arguments

`template.ranges`
: A ‘GRanges‘ object with genomic locations (same object must be supplied to this and to the ‘simulateData‘ functions).

`nsamples`
: A single integer >= 1 indicating the number of samples to which AMRs will be assigned.

`exclude.ranges`
: A ‘GRanges‘ object with genomic locations. None of the simulated AMRs in the output will overlap with any of regions from ‘exclude.ranges‘. If ‘NULL‘ (the default), AMRs are not restricted by their genomic location.

`regions.per.sample`
: A single integer >= 1 (the default). Number of AMRs to be assigned to every sample. Message is shown and the ‘regions.per.sample‘ value is limited to ‘maxnAMR (where ‘maxnAMR‘ is the maximum number of potential AMRs for the ‘template.ranges‘).

`samples.per.region`
: A single integer >= 1 (the default). Number of samples to which the same AMR will be assigned. Message is shown and the ‘samples.per.region‘ value is limited to ‘nsamples‘ if the former is greater than the latter.

`sample.names`
: A character vector with sample names. If ‘NULL‘ (the default), sample names will be computed as ‘paste0("sample", seq_len(nsamples))‘. When specified, the length of the ‘sample.names‘ vector must not be smaller than the value of ‘nsamples‘.

`merge.window`
: A positive integer. All ‘template.ranges‘ genomic locations within this distance will be merged to create a list of potential AMRs (which will be later filtered from regions overlapping with any regions from the ‘exclude.ranges‘).

`min.cpgs`
: A single integer >= 1. All AMRs containing less than ‘min.cpgs‘ genomic locations are filtered out. The default: 7.

`max.cpgs`
: A single integer >= 1. All AMRs containing more than ‘max.cpgs‘ genomic locations are filtered out. The default: ‘Inf‘.

`min.width`
: A single integer >= 1 (the default). Only AMRs with the width of at least ‘min.width‘ are returned.

`dbeta`
: A single non-negative numeric value in the range [0,1] or a numeric vector of such values (with as many elements as there are AMRs). Used to populate the ‘dbeta‘ metadata column, defines a desired absolute deviation of corresponding AMR from the median for the ‘simulateData‘ function.

Details

Using provided template (‘GRanges‘ object) ‘simulateAMR‘ randomly selects genomic regions satisfying various criteria (number of CpGs, width of the region) and assigns them to samples according to specified parameters (number of AMRs per sample, number of samples per AMR). Its output is meant to be used as the set of true positive AMRs for the ‘simulateData‘ function.

Value

The output is a ‘GRanges‘ object that contains a subset of aberrantly methylated regions (AMRs) randomly selected from all the possible AMRs for the provided ‘template.ranges‘ object. The following metadata columns are included:

- ‘revmap‘ – integer list of ‘template.ranges‘ genomic locations that are included in this AMR region
- ‘ncpg‘ – number of ‘template.ranges‘ genomic locations within this AMR region
- ‘sample‘ – an identifier of a sample to which corresponding AMR belongs
- ‘dbeta‘ – equals to supplied ‘dbeta‘ parameter

See Also

simulateData for the generation of simulated test data sets, getAMR for identification of AMRs, plotAMR for plotting AMRs, getUniverse for info on enrichment analysis, and ‘ramr‘ vignettes for the description of usage and sample data.

Examples

```
data(ramr)
set.seed(1)
amrs.unique <-
  simulateAMR(ramr.data, nsamples=4, regions.per.sample=2,
              min.cpgs=5, merge.window=1000, dbeta=0.2)
amrs.nonunique <-
  simulateAMR(ramr.data, nsamples=3, exclude.ranges=amrs.unique,
              samples.per.region=2, min.cpgs=5, merge.window=1000)
```

---

simulateData *Template-based simulation of methylation data sets*

Description

‘simulateData‘ generates aberration-free methylation data using an experimental data set as a template, and further introduces methylation aberrations if ‘GRanges‘ object containing a set of aberrantly methylated regions was provided. The output can be used to evaluate performance of algorithms for search of differentially (DMR) or aberrantly (AMR) methylated regions.

Usage

```
simulateData(
  template.ranges,
  nsamples,
  amr.ranges = NULL,
  sample.names = NULL,
  compute = "beta+binom",
  compute.estimate = c("mom", "amle", "nmle"),
  compute.weights = c("equal", "logInvDist", "sqrtInvDist", "invDist"),
  ncores = NULL,
  verbose = TRUE
)
```

Arguments

`template.ranges`
: A ‘GRanges‘ object with genomic locations and corresponding methylation beta values included as metadata (the same object must be supplied to this and to the ‘simulateAMR‘ functions).

`nsamples`
: A single integer >= 1 indicating the number of samples to generate.

`amr.ranges` A ‘GRanges‘ object with genomic locations of methylation aberrations (epimutations). If ‘NULL‘ (the default), no aberrations is introduced, and function will return "smoothed" data set. If supplied, ‘GRanges‘ object must contain the following metadata columns:

- ‘revmap‘ – integer list of ‘template.ranges‘ genomic locations that are included in this AMR region
- ‘sample‘ – an identifier of a sample to which corresponding AMR belongs. Must be among the supplied or auto generated ‘sample.names‘
- ‘dbeta‘ – absolute deviation to be introduced. Must be numeric within the closed interval [0,1] or NA. When NA - the resulting beta value for the corresponding genomic position will also be NA

Such an object can be obtained using `simulateAMR` method or manually.

`sample.names` A character vector with sample names. If ‘NULL‘ (the default), sample names will be auto generated. When specified, the length of the ‘sample.names‘ vector must be equal to the value of ‘nsamples‘.

`compute` A single string for the distribution to fit to the data. Currently accepts "beta+binom" (the default) only, which stands for endpoint-inflated beta distribution. See Details section and `getAMR` method description for additional explanations.

`compute.estimate` A single string for the method of parameter estimation of beta distribution. The default ("mom") stands for the method of moments based on the unbiased estimator of variance and includes {0;1} endpoints in calculation of moments (mean, unbiased variance). Other options are "amle" (approximation of maximum likelihood estimation) and "nmle" (numeric maximum likelihood estimation) - both ignore {0;1} endpoints in calculations. More details on these methods are given in `getAMR` method description.

`compute.weights` A single string for the method to compute optional sample weights that are used during estimation of beta distribution parameters. If default ("equal"), all weights are equal. Otherwise, weight of a value equals to a natural logarithm of inverse absolute distance of this value to the sample median ("logInvDist"), a square root of inverse absolute distance of this value to the sample median ("sqrtInvDist"), or an inverse absolute distance of this value to the sample median ("invDist"). Using weighted parameter estimation allows to increase sensitivity of outlier detection. More details on weighted parameter estimation are given in `getAMR` method description.

`ncores` A single integer >= 1 for the number of OpenMP threads for parallel computation. By default (NULL), function will use half of available cores. When the same random seed is set, results of this function are always identical (reproducible), even when more than one core is used (at a cost of serial random number generation).

`verbose` boolean to report progress and timings (default: TRUE).

Details

For every genomic location in the template data set (‘GRanges‘ object with genomic locations and corresponding methylation beta values included as metadata) ‘simulateData‘ does the following:

- estimates parameters of beta distribution
- in the same input, calculates frequencies of zero and one values (endpoints; whenever present)
- uses estimated parameters of beta distribution and probabilities (observed frequencies) of {0;1} values to generate ‘nsamples‘ random values by means of ‘stats::rbeta‘ function (for beta values) and/or ‘stats::rbinom‘ function (for {0;1} endpoint values, according to their frequencies and therefore probabilities).

This results in "smoothed" data set that has biologically relevant distribution of methylation values at every genomic location, but does not contain methylation aberrations. If the ‘amr.ranges‘ parameter points to a ‘GRanges‘ object with aberrations, every AMR is then introduced into the "smoothed" data set as following: if mean methylation beta value for AMR region across all samples in the "smoothed" data set is above (below) 0.5 then all beta values for the sample defined by the ‘sample‘ metadata column are decreased (increased) by the absolute value specified in the ‘dbeta‘ metadata column. Resulting data sets with (or without) AMR together with the ‘amr.ranges‘ set of true positive aberrations can be used as test data set to evaluate performance of algorithms for search of differentially (DMR) or aberrantly (AMR) methylated regions.

Value

The output is a ‘GRanges‘ object with genomic ranges that are equal to the genomic ranges of the provided template and metadata columns containing generated methylation beta values for ‘nsamples‘ samples. If ‘amr.ranges‘ object was supplied, then randomly generated beta values will be modified accordingly.

Note

NA values within metadata columns of ‘template.ranges‘ are silently dropped in all computations. NA values will also not appear in the result of this function, unless parameters of beta distribution and/or probabilities of zeros or ones cannot be estimated (e.g., due to too many NA values in ‘template.ranges‘ metadata).

See Also

simulateAMR for the generation of random methylation aberrations, getAMR for identification of AMRs, plotAMR for plotting AMRs, getUniverse for info on enrichment analysis, and ‘ramr‘ vignettes for the description of usage and sample data.

Examples

```
data(ramr)
set.seed(1)
amrs <-
  simulateAMR(ramr.data, nsamples=10, regions.per.sample=3,
              samples.per.region=1, min.cpgs=5, merge.window=1000)
noise <-
  simulateAMR(ramr.data, nsamples=10, regions.per.sample=20,
              exclude.ranges=amrs, min.cpgs=1, max.cpgs=1, merge.window=1)
noisy.data <-
  simulateData(template.ranges=ramr.data, nsamples=10, amr.ranges=c(amrs,noise))
plotAMR(data.ranges=noisy.data, amr.ranges=amrs[1])
```

simulateData.obsolete *[OBSOLETE] Template-based simulation of methylation data sets*

Description

This function is fully functional but obsolete. It will remain a part of the package for consistency, as it was used in ‘ramr‘ publication (doi:10.1093/bioinformatics/btab586). Please use faster and more capable `simulateData` instead.

‘simulateData.obsolete‘ generates aberration-free methylation data using an experimental data set as a template, and further introduces methylation aberrations if ‘GRanges‘ object containing a set of aberrantly methylated regions was provided. The output can be used to evaluate performance of algorithms for search of differentially (DMR) or aberrantly (AMR) methylated regions.

Usage

```
simulateData.obsolete(
  template.ranges,
  nsamples,
  amr.ranges = NULL,
  sample.names = NULL,
  min.beta = 0.001,
  max.beta = 0.999,
  cores = max(1, parallel::detectCores() - 1),
  verbose = TRUE
)
```

Arguments

`template.ranges`
: A ‘GRanges‘ object with genomic locations and corresponding beta values included as metadata (same object must be supplied to this and to the ‘simulateAMR‘ functions).

`nsamples`
: A single integer >= 1 indicating the number of samples to generate.

`amr.ranges`
: A ‘GRanges‘ object with genomic locations of (rare) methylation aberrations. If ‘NULL‘ (the default), no aberrations is introduced, and function will return "smoothed" data set. If supplied, ‘GRanges‘ object must contain the following metadata columns:

  - ‘revmap‘ – integer list of ‘template.ranges‘ genomic locations that are included in this AMR region
  - ‘sample‘ – an identifier of a sample to which corresponding AMR belongs. Must be among the supplied or auto generated ‘sample.names‘
  - ‘dbeta‘ – absolute deviation to be introduced. Must be numeric within the range c(0,1) or NA. When NA - the resulting beta value for the corresponding genomic position will also be NA

  Such an object can be obtained using `simulateAMR` method or manually.

sample.names	A character vector with sample names. If ‘NULL‘ (the default), sample names will be computed as ‘paste0("sample", seq_len(nsamples))‘. When specified, the length of the ‘sample.names‘ vector must be equal to the value of ‘nsamples‘.
min.beta	A single numeric within the range c(0,1). All beta values in the generated data set below this value will be assigned this value. The default: 0.001.
max.beta	A single numeric within the range c(0,1). All beta values in the generated data set above this value will be assigned this value. The default: 0.999.
cores	A single integer >= 1. Number of processes for parallel computation (the default: all but one cores). Results of parallel processing are fully reproducible when the same seed is used (thanks to doRNG).
verbose	boolean to report progress and timings (default: TRUE).

Details

For every genomic location in the template data set (‘GRanges‘ object with genomic locations and corresponding beta values included as metadata) ‘simulateData.obsolete‘ estimates the parameters of beta distribution by means of ‘EnvStats::ebeta‘ function, and then uses estimated parameters to generate ‘nsamples‘ random beta values by means of ‘stats::rbeta‘ function. This results in "smoothed" data set that has biologically relevant distribution of beta values at every genomic location, but does not contain methylation aberrations. If the ‘amr.ranges‘ parameter points to a ‘GRanges‘ object with aberrations, every AMR is then introduced into the "smoothed" data set as following: if mean methylation beta value for AMR region across all samples in the "smoothed" data set is above (below) 0.5 then all beta values for the sample defined by the ‘sample‘ metadata column are decreased (increased) by the absolute value specified in the ‘dbeta‘ metadata column. Resulting data sets with (or without) AMR together with the ‘amr.ranges‘ set of true positive aberrations can be used as test data set to evaluate performance of algorithms for search of differentially (DMR) or aberrantly (AMR) methylated regions.

Value

The output is a ‘GRanges‘ object with genomic ranges that are equal to the genomic ranges of the provided template and metadata columns containing generated methylation beta values for ‘nsamples‘ samples. If ‘amr.ranges‘ object was supplied, then randomly generated beta values will be modified accordingly.

See Also

simulateAMR for the generation of random methylation aberrations, getAMR for identification of AMRs, plotAMR for plotting AMRs, getUniverse for info on enrichment analysis, and ‘ramr‘ vignettes for the description of usage and sample data.

Examples

```
data(ramr)
amrs <-
  simulateAMR(ramr.data, nsamples=10, regions.per.sample=3,
              samples.per.region=1, min.cpgs=5, merge.window=1000)
noise <-
  simulateAMR(ramr.data, nsamples=10, regions.per.sample=20,
```

```
              exclude.ranges=amrs, min.cpgs=1, max.cpgs=1, merge.window=1)
noisy.data <-
  simulateData.obsolete(ramr.data, nsamples=10, amr.ranges=c(amrs,noise),
                        cores=2)
plotAMR(noisy.data, amr.ranges=amrs[1])
```

Index

∗ **data**
    ramr.data, 13
∗ **sets**
    ramr.data, 13

getAMR, 2, *8*, *11*, *13*, *16–18*, *20*
getAMR.obsolete, 8
getUniverse, *7*, *10*, 10, *13*, *16*, *18*, *20*

plotAMR, *7*, *10, 11*, 11, *16*, *18*, *20*

ramr.data, 13
ramr.samples *(*ramr.data*)*, 13
ramr.tp.nonunique *(*ramr.data*)*, 13
ramr.tp.unique *(*ramr.data*)*, 13

simulateAMR, *7*, *10, 11*, *13*, 14, *17–20*
simulateData, *7*, *10, 11*, *13*, *16*, 16, *19*
simulateData.obsolete, 19