Package 'ramr'

May 2, 2024

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

Version 1.13.0

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

Description ramr is an R package for detection of low-frequency aberrant methylation events in large data sets obtained by methylation profiling using array or high-throughput bisulfite 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.

Depends R (>= 4.1), GenomicRanges, parallel, doParallel, foreach, doRNG, methods

Imports IRanges, BiocGenerics, ggplot2, reshape2, EnvStats, ExtDist, matrixStats, S4Vectors

Suggests RUnit, knitr, rmarkdown, gridExtra, annotatr, LOLA, org.Hs.eg.db, TxDb.Hsapiens.UCSC.hg19.knownGene

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

RoxygenNote 7.1.2

VignetteBuilder knitr

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

git_branch devel

git_last_commit 751437b

git_last_commit_date 2024-04-30

Repository Bioconductor 3.20

Date/Publication 2024-05-01

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

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

Contents

`getAMR` *Search for aberrantly methylated regions*

Description

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

Usage

```
getAMR(
  data.ranges,
  data.samples = NULL,
  ramr.method = "IQR",
  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" or "wbeta" - filtering based on fitting non-weighted (‘EnvStats::ebeta‘) or weighted (‘ExtDist::eBeta‘) 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‘ compares methylation beta values of each sample with other samples to identify rare long-range methylation aberrations. 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=="*beta"‘: parameters of beta distribution are estimated by means of ‘EnvStats::ebeta‘ or ‘ExtDist::eBeta‘ functions, and 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(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(ramr.data, ramr.samples, ramr.method="beta", min.cpgs=5,
               merge.window=1000, qval.cutoff=1e-3, cores=2)

# 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,
  data.samples = NULL,
  amr.ranges,
  highlight = NULL,
  title = NULL,
  window = 300
)
```

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) to be included in the plot. If 'NULL' (the default), then all samples (metadata columns) are included.
amr.ranges	An output of 'getAMR' - a 'GRanges' object that contain aberrantly methylated regions (AMRs).
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.
window	An optional integer constant to expand genomic ranges of the 'amr.ranges' object (the default: 300).

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(ramr.data, ramr.samples, ramr.tp.unique[1])
library(gridExtra)
do.call("grid.arrange",
        c(plotAMR(ramr.data, ramr.samples, 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(ramr.data, ramr.samples, ramr.method="beta", min.cpgs=5,
               merge.window=1000, qval.cutoff=1e-3, cores=2)
plotAMR(ramr.data, ramr.samples, amrs[1])
plotAMR(ramr.data, ramr.samples, 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)
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,
  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‘ 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(ramr.data, nsamples=10, amr.ranges=c(amrs,noise), cores=2)
plotAMR(noisy.data, amr.ranges=amrs[1])
```

Index