Package ‘RnBeads’

April 1, 2025

Title RnBeads

Description RnBeads facilitates comprehensive analysis of various types of DNA methylation data at the genome scale.

Date 2024-04-22

Version 2.24.0

Suggests Category, GOstats, Gviz, IlluminaHumanMethylation450kmanifest, RPMM, RnBeads.hg19, RnBeads.mm9, RnBeads.hg38, XML, annotate, biomaRt, foreach, doParallel, ggbio, isva, mclust, mgcv, minfi, nlme, org.Hs.eg.db, org.Mm.eg.db, org.Rn.eg.db, quadprog, rtracklayer, qvalue, sva, wateRmelon, wordcloud, qvalue, argparse, glmnet, IlluminaHumanMethylation450kanno.ilmn12.hg19, scales, missMethyl, impute, shiny, shinyjs, plotrix, hexbin, RUnit, MethylSeekR, sesame

Depends R (>= 3.0.0), BiocGenerics, S4Vectors (>= 0.9.25), GenomicRanges, MASS, cluster, ff, fields, ggplot2 (>= 0.9.2), gplots, grid, gridExtra, limma, matrixStats, methods, illuminaio, methylumi, plyr

Imports IRanges

License GPL-3

biocViews DNAMethylation, MethylationArray, MethylSeq, Epigenetics, QualityControl, Preprocessing, BatchEffect, DifferentialMethylation, Sequencing, CpGIsland, ImmunoOncology, TwoChannel, DataImport

Collate 'CNV.R' 'Report-class.R' 'Report-methods.R' 'ReportPlot-class.R' 'ReportPlot-methods.R' 'RnBDiffMeth-class.R' 'bigFf.R' 'RnBSet-class.R' 'RnBeadSet-class.R' 'RnBeadRawSet-class.R' 'RnBeads-package.R' 'RnBiseqSet-class.R' 'agePrediction.R' 'annotations.R' 'batch.R' 'batch.quality.R' 'bmiq.R' 'cellTypeAdjustment.R' 'clusterArchitecture.R' 'clusterArchitectureLSF.R' 'clusterArchitectureSGE.R' 'clusterArchitectureSLURM.R' 'clustering.R' 'combining.R' 'computeCluster.R' 'controlPlots.R' 'controlPlotsBiSeq.R' 'dataExport.R' 'dataImport.R' 'differentialMethylation.R' 'differentialVariability.R' 'enmix.R' 'enrichment.R' 'exportGEO.R' 'filtering.R' 'filteringSummary.R' 'genomewide.R' 'greedycut.R' 'immune.R' 'loading.R' 'logger.R' 'lolaUtils.R' 'main.R' 'normalization.R' 'normalizationMethods.R' 'options.R' 'parallelProcessing.R' 'plottingUtils.R' 'profiles.R' 'qualityControl.R' 'readGEO.R' 'regionDescription.R' 'regionProfiles.R' 'segmentation.R' 'sesame.R' 'sex.R' 'subSegments.R' 'sva.R' 'utilities.R' 'wbcInference.R'

RoxygenNote 7.3.1

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

git_branch RELEASE_3_20

git_last_commit a6b1698

git_last_commit_date 2024-10-29

Repository Bioconductor 3.20

Date/Publication 2025-03-31

Author Yassen Assenov [aut],
Christoph Bock [aut],
Pavlo Lutsik [aut],
Michael Scherer [aut],
Fabian Mueller [aut, cre]

Maintainer Fabian Mueller `<team@rnbeads.org>`

Contents

`accepted` *RnBeads option values and restrictions*

Description

The values of options in RnBeads are stored in dedicated R objects accompanying the package. These objects are named `infos`, `accepted`, `current` and `previous`. They should not be loaded or otherwise operated on by users. Please refer to the documentation of `rnb.options` for accessing and modifying option values in **RnBeads**.

Format

`infos` is a `data.frame` containing information about all options in **RnBeads**. Row names in this table are the option names; the column names are `"Type"`, `"Named"`, `"Null"`, `"Max"`, `"Min"`, `"MaxInclusive"` and `"MinInclusive"`. `accepted` is a `list` containing the sets of accepted values for some of the options. `current` is a `list` with current values for all options. `previous` is a `list` with previous values for the affected options; this list is only temporarily used while setting option values through `rnb.options` or `rnb.xml2options`.

Author(s)

Yassen Assenov

`addDiffMethTable,RnBDiffMeth-method` *addDiffMethTable-methods*

Description

Adds a differential methylation table

Usage

```
## S4 method for signature 'RnBDiffMeth'
addDiffMethTable(
  object,
  dmt,
  comparison,
  region.type,
  grp.labs = c("group1", "group2")
)
```

Arguments

`object`	`RnBDiffMeth` object
`dmt`	Differential methylation table to add
`comparison`	character or index of the comparison of the table to retrieve
`region.type`	character or index of the region type of the table to retrieve
`grp.labs`	character vector of length 2 specifying the names of the groups being compared

Value

the updated RnBDiffMeth object

Note

Caveat: if disk dumping is enabled the resulting object tables will be stored in the initial location of the object.

Author(s)

Fabian Mueller

Examples

```
library(RnBeads.hg19)
data(small.example.object)
logger.start(fname=NA)
dm <- rnb.execute.computeDiffMeth(rnb.set.example,"Sample_Group",c("genes","tiling"))
s.groups <- rnb.sample.groups(rnb.set.example,"Sample_Group")[[1]]
dmt.sites <- computeDiffTab.extended.site(meth(rnb.set.example),s.groups[[1]],s.groups[[2]])
map.regions.to.sites <- regionMapping(rnb.set.example,"promoters")
dmt.promoters <- computeDiffTab.default.region(dmt.sites,map.regions.to.sites)
cmp.name <- get.comparisons(dm)[1]
grp.labs <- get.comparison.grouplabels(dm)[1,]
#add the promoter level differential methylation table
dm.add <- addDiffMethTable(dm,dmt.promoters,cmp.name,"promoters",grp.labs)
get.region.types(dm.add)
```

`addPheno,RnBSet-method` *addPheno*

Description

Adds phenotypic or processing information to the sample annotation table of the given `RnBSet` object.

Usage

```
## S4 method for signature 'RnBSet'
addPheno(object, trait, header)
```

Arguments

`object`	`RnBSet` of interest.
`trait`	Trait as a non-empty `vector` or `factor`. The length of this vector must be equal to the number of samples in `object`, the i-th element storing the value for the i-th sample. Note that names, if present, are ignored.
`header`	Trait name given as a one-element `character`. This is the heading to be used for the sample annotation table. This method fails if such a trait already exists; in other words, if `header %in% names(pheno(object))`.

Value

The modified dataset as an object of type `RnBSet`.

Author(s)

Fabian Mueller

Examples

```
library(RnBeads.hg19)
data(small.example.object)
logger.start(fname=NA)
is.hiPSC <- pheno(rnb.set.example)[, "Sample_Group"]=="hiPSC"
rnb.set.mod <- addPheno(rnb.set.example, is.hiPSC, "is_hiPSC")
pheno(rnb.set.mod)
```

`addRegionSubsegments` *addRegionSubsegments*

Description

For the region annotation of a given `RnBSet` object. Subdivide each region into subsegments by hierarchical clustering on the site distances in a particular region and then splitting the region into subregions consisting of these site clusters. The number of clusters is determined in such way that the mean number of sites per cluster is given by the `ns` parameter.

Usage

```
addRegionSubsegments(
  rnb.set,
  annotation.dir,
  region.types = NULL,
  add.region.types.to.options = FALSE,
  ns = 10
)
```

Arguments

`rnb.set`	an `RnBSet` `object`
`annotation.dir`	a directory to save the annotation to for later reloading. (binary `RData` format.)
`region.types`	the region types to which subsegmentation should be applied. Must be a non-empty subset of `summarized.regions(rnb.set)`. Defaults (`NULL`) to all region types in `rnb.set`
`add.region.types.to.options`	Flag indicating whether to add the newly created subregions to the package's `region.types` option
`ns`	the mean number of sites per cluster.

Value

the modified `RnBSet` object

Author(s)

Fabian Mueller

Examples

```
library(RnBeads.hg19)
data(small.example.object)
logger.start(fname=NA)
rnb.set.mod <- addRegionSubsegments(rnb.set.example,tempdir(),region.types=c("tiling","genes"))
summary(meth(rnb.set.mod,type="tiling.subsegments"))
```

`annotation,RnBSet-method`
annotation-methods

Description

Genomic annotation of the methylation sites or regions covered in the supplied dataset.

Usage

```
## S4 method for signature 'RnBSet'
annotation(object, type = "sites", add.names = FALSE, include.regions = FALSE)
```

Arguments

`object`	dataset as an object of type inheriting `RnBSet`.
`type`	loci or regions for which the annotation should be obtained. If the value of this parameter is `"sites"` (default), individual methylation sites are annotated. Otherwise, this must be one of the available region types, as returned by `rnb.region.types`.
`add.names`	flag specifying whether the unique site identifiers should be used as row names of the resulting data frame
`include.regions`	if `TRUE` one additional column is added to the returned annotation dat frame for each of the available region types, giving the indices of the

Value

Annotation table in the form of a `data.frame`.

Author(s)

Pavlo Lutsik

Examples

```
library(RnBeads.hg19)
data(small.example.object)
## show present sites
head(annotation(rnb.set.example, add.names=TRUE))
## show promoters
ann.prom<-annotation(rnb.set.example, type="promoters", add.names=TRUE)
head(ann.prom)
```

`apply.iEVORA` *apply.iEVORA*

Description

This routine applies the iEVORA method created by Teschendorff et.al. to the supplied methylation matrix in a similar way as the diffVar method.

Usage

```
apply.iEVORA(meth.matrix, inds.g1, inds.g2)
```

Arguments

`meth.matrix`	Matrix containing the methylation information used to calculate differentially variable sites between the two groups
`inds.g1`	Indices in the phenotypic table corresponding to the first group.
`inds.g2`	Indices in the phenotypic table corresponding to the second group.

Value

Q-values as the result of applying the iEVORA method and then correct for multiple testing.

Author(s)

Michael Scherer

`as.RnBeadRawSet` *Conversion to/from RnBeadRawSet*

Description

The `"as"` method can be used for the following conversions:

- `MethyLumiSet` (in package **methylumi**) to `RnBeadRawSet`
- `RnBeadRawSet` to `MethyLumiSet`
- `RGChannelSet` (in package **minfi**) to `RnBeadRawSet`

`assembly,RnBSet-method` *assembly-methods*

Description

Extracts information about assembly

Usage

```
## S4 method for signature 'RnBSet'
assembly(object)
```

Arguments

`object`	Dataset of interest.

Value

Sample annotation information available for the dataset in the form of a `data.frame`.

Examples

```
library(RnBeads.hg19)
data(small.example.object)
assembly(rnb.set.example) # "hg19"
```

`auto.select.rank.cut` *auto.select.rank.cut*

Description

automatically select a rank cutoff for given ranks and p-values current implementation: sort the p-values according to rank. select as rank cutoff the rank for which the worst (i.e. max) p-value in the top list is still smaller than the best (i.e. min) p-value of the group of worst-ranking p-values of equal size as the top-list

Usage

```
auto.select.rank.cut(p, r, alpha = 0.1)
```

Arguments

`p`	vector of p-values
`r`	vector of ranks
`alpha`	the percentile to select the top and bottom part of the list

Value

the maximum rank fulfilling the criterion

Author(s)

Fabian Mueller

`BMIQ` *BMIQ*

Description

This function makes 3 indipendent attempts to fit a 3-state beta mixture model on the provided type I probes. An attempt is successful if at least 4 probes are assigned to each level. In case all attempts fail, the return value is `NULL`.

Usage

```
BMIQ(
  beta.v,
  design.v,
  doH = TRUE,
  nfit = 50000,
  th1.v = c(0.2, 0.75),
  th2.v = NULL,
  niter = 5,
  tol = 0.001
)
```

Arguments

`beta.v`	`double` vector consisting of beta values. Missing values (`NA`s) cannot be handled, so these must be removed or imputed prior to running BMIQ. Before normalization, beta values that are exactly 0 and exactly 1 are replaced by the minimum positive and maximum value below 1, respectively.
`design.v`	`integer` vector of length `length(beta.v)`, containing the values `1` and `2` only. These values specify probe design type.
`doH`	Flag indicating if normalization for hemimethylated type II probes is to be performed.
`nfit`	Number of probes of a given design to use for the fitting. Smaller values will make BMIQ faster at the expense of accuracy. Values between 10000 and 50000 seem to work well.
`th1.v`	Thresholds "type 1" to use for the initialization of the EM algorithm. These values should represent best guesses for calling type I probes hemi-methylated and methylated, and are refined in further steps by the algorithm.
`th2.v`	Thresholds "type 2" to used for the initialization of the EM algorithm. These values should represent best guesses for calling type II probes hemi-methylated and methylated, and are refined in further steps by the EM algorithm. If this is `NULL` (default), the thresholds are estimated based on `th1.v` and a modified PBC correction method.
`niter`	Maximum number of EM iterations to be performed.
`tol`	Tolerance threshold for EM algorithm.

Details

Performs Beta-mixture quantile normalization, adjusting for type II bias in Infinium 450K data.

Value

List with the following elements:

`"all"` The normalised beta-profile for the sample.

`"class1"` Methylation state assigned to the type I probes.

`"class2"` Methylation state assigned to the type II probes.

`"av1"` Mean beta values for the `nL` classes for type I probes.

`"av2"` Mean beta values for the `nL` classes for type II probes.

`"hf"` Hubble dilation factor.

`"th1"` Estimated thresholds used for type I probes.

`"th2"` Estimated thresholds used for type II probes.

Author(s)

Andrew Teschendorff and Steve Horvath; with minor modifications by Yassen Assenov

---

`ClusterArchitecture-class` *ClusterArchitecture Class*

Description

A virtual class for storing specifications of architectures for different compute clusters. It is designed to let other classes inherit from it

Details

For a concrete child class for a sun grid architecture specification see `ClusterArchitectureSGE` If you want to implement your own child class be sure to at least implement the following functions: `getSubCmdTokens,ClusterArchitecture-method`.

Slots

`name` A name or identifier

`executables` A NAMED character vector of executables that can be used by the cluster. For instance, the `R` executable is important

`getSubCmdTokens.optional.args` character vector containing the valid optional arguments to the `getSubCmdTokens,ClusterArchitecture-method` function.

Methods

`getSubCmdTokens,ClusterArchitecture-method` Returns a vector of command line tokens corresponding to submitting a job with the given command to the cluster

`getSubCmdStr,ClusterArchitecture-method` Returns a string for the of command line corresponding to submitting a job with the given command to the cluster

`setExecutable,ClusterArchitecture,character,character-method` Tells the cluster architecture about an executable that can be submitted as job

`getExecutable,ClusterArchitecture,character-method` Gets the location of an executable associated with a name

Author(s)

Fabian Mueller

`ClusterArchitectureLSF-class` *ClusterArchitectureLSF Class*

Description

A child class of `ClusterArchitecture` implementing specifications of IBM LSF architectures.

Details

Follow this template if you want to create your own ClusterArchitecture class.

Slots

see `ClusterArchitecture`

Methods

`getSubCmdTokens,ClusterArchitectureLSF-method` Returns a vector of command line tokens corresponding to submitting a job with the given command to the cluster

Author(s)

Michael Scherer

`ClusterArchitectureSGE-class`

ClusterArchitectureSGE Class

Description

A child class of `ClusterArchitecture` implementing specifications of Sun Grid Engine (SGE) architectures.

Details

Follow this template if you want to create your own ClusterArchitecture class.

Slots

see `ClusterArchitecture`

Methods

`getSubCmdTokens,ClusterArchitectureSGE-method` Returns a vector of command line tokens corresponding to submitting a job with the given command to the cluster

Author(s)

Fabian Mueller

`ClusterArchitectureSLURM-class`

ClusterArchitectureSLURM Class

Description

A child class of `ClusterArchitecture` implementing specifications of Simple Linux Utility for Resource Management (SLURM) architectures.

Details

Follow this template if you want to create your own ClusterArchitecture class.

Slots

see `ClusterArchitecture`

Methods

`getSubCmdTokens,ClusterArchitectureSGE-method` Returns a vector of command line tokens corresponding to submitting a job with the given command to the cluster

Author(s)

Michael Scherer

`coercion-methods` *as("RnBeadSet", "MethyLumiSet")*

Description

Convert a `RnBeadSet` object to `MethyLumiSet`

Convert a `RnBeadSet` object to a "mock" `RnBiseqSet` object (used in the combine method)

`cols.to.rank.site` *cols.to.rank.site*

Description

Return a matrix containing the negative absolute values of the information used to rank the sites. Those are currently: the variance difference, the log ratio in variances and the p-value from the statistical test.

Usage

```
cols.to.rank.site(diff.var)

cols.to.rank.region(diff.var)
```

Arguments

`diff.var`	A differential variability table.

Value

A matrix with the absolute values of the relevant columns

Author(s)

Michael Scherer

`combine.diffMeth.objs` *combine.diffMeth.objs*

Description

combine differential methylation objects (output from `rnb.run.differential`). To be more precise, the `diffmeth` and `dm.go.enrich` are merged. individual objects that are merged are assumed to belong to the same analysis and vary only in their indexing of region types and comparisons

Usage

```
combine.diffMeth.objs(obj.list)
```

Arguments

`obj.list`	a list containing outputs from `rnb.run.differential`

Author(s)

Fabian Mueller

`combine.rnb.sets,RnBSet,RnBSet-method` *combine.rnb.sets-methods*

Description

Combine two objects inheriting from `RnBSet` class

Usage

```
## S4 method for signature 'RnBSet,RnBSet'
combine.rnb.sets(x, y, type = "all")
```

Arguments

`x, y`	`RnBeadSet`, `RnBeadRawSet` or `RnBiseqSet` object
`type`	`character` singleton defining the set operation applied to the two site sets, one of "all", "all.x", "all.y" or "common"

Details

Combine method supports a merge of any two RnBSet objects that contain data of the same specie. In case a non-synonymous merge is performed, the class conversion will follow the following hierarchy: `RnBeadSet` < `RnBeadRawSet` < `RnBiseqSet`. In case `x` and `y` are both array data containers (`RnBeadSet` or `RnBeadRawSet`), the resulting object will have an annotation that corresponds to the newer array version (`27k` < `450k` < `EPIC` < `EPICv2`). The sample sets of `x` and `y` should be unique. Sample annotation information is merged only for columns which have identical names in both objects. CpG sites of the new object are a union of those present in both objects.

Value

combined `RnBeadSet`, `RnBeadRawSet` or `RnBiseqSet` object

Examples

```
library(RnBeads.hg19)
data(small.example.object)
r1 <- rnb.set.example
r1 <- remove.samples(r1,samples(rnb.set.example)[1:5])
i <- which(r1@sites[,2] == 15 | r1@sites[,2] == 21)
sites.rem.r1 <- union(sample(1:nrow(meth(rnb.set.example)),500),i)
r1 <- remove.sites(r1,sites.rem.r1)
r2 <- rnb.set.example
r2 <- remove.samples(r2,samples(rnb.set.example)[6:12])
sites.rem.r2 <- sample(1:nrow(meth(rnb.set.example)),800)
r2 <- remove.sites(r2,sites.rem.r2)
rc <- combine.rnb.sets(r1,r2)
#assertion: check the number of sites
sites.rem.c <- intersect(sites.rem.r1,sites.rem.r2)
(nrow(meth(rnb.set.example))-length(sites.rem.c)) == nrow(meth(rc))
```

`combineTestPvalsMeth` *combineTestPvalsMeth*

Description

combine p-values of multiple tests using (a generalization of) Fisher's method. The parameter setting here is taylored to DNA methylation, but can be adapted. Reference: Makambi, K. (2003). Weighted inverse chi-square method for correlated significance tests. Journal of Applied Statistics, 30(2), 225-234.

Usage

```
combineTestPvalsMeth(
  pvalues,
  testWeights = NULL,
  correlated = FALSE,
  methExpectedTestCorrelation = 0.8
)
```

Arguments

`pvalues`	p-values to combine
`testWeights`	weights for the individual tests
`correlated`	are the individual tests correlated
`methExpectedTestCorrelation`	expected correlation. Empirically approximated to the default value of 0.8 for DNA-methylation

Value

the combined p-value

Author(s)

Fabian Mueller, Christoph Bock

Examples

```
p.vals <- 10^-c(0,1,5)
combineTestPvalsMeth(p.vals)
```

`computeDiffTab.default.region` *computeDiffTab.region*

Description

computes a difference table containing multiple difference measures, In the simple version the mean of the difference in means, the mean quotient in means and a combination of p-values on the site level are computed. This is computed for each row of the input table. The extended version contains additional columns

Usage

```
computeDiffTab.default.region(dmtp, regions2sites, includeCovg = FALSE)
```

Arguments

`dmtp`	differential methylation table on the site level (as obtained from `computeDiffTab.default.site`)
`regions2sites`	a list containing for each region the indices of the corresponding sites in the site differential methylation table
`includeCovg`	flag indicating whether to include coverage information

Value

a dataframe containing the following variables for a given genomic region:

`mean.mean.g1`, `mean.mean.g2`	mean of mean methylation levels for group 1 and 2 across all sites in a region
`mean.mean.diff`	Mean difference in means across all sites in a region
`mean.mean.quot.log2`	Mean quotient in means across all sites in a region
`comb.p.val`	Combined p-value using a generalization of Fisher's method. See `combineTestPvalsMeth` for details.
`comb.p.adj.fdr`	FDR adjusted combined p-value
`num.sites`	number of sites that were considered for a region
`mean.num.na.g1/2`	mean number (accross all considered sites) of samples that contained an NA for group 1 and 2 respectively
`mean.mean.covg.g1/2`	Mean value of mean coverage values (across all samples in a group) across all sites in a region
`mean.nsamples.covg.thresh.g1/2`	mean number (accross all considered sites) of samples that have a coverage larger than the specified threshold (see `computeDiffTab.default.site` for details) for group 1 and 2 respectively

Author(s)

Fabian Mueller

Examples

```
library(RnBeads.hg19)
data(small.example.object)
logger.start(fname=NA)
meth.mat <- meth(rnb.set.example)
sample.groups <- rnb.sample.groups(rnb.set.example)[[1]]
dm.sites <- computeDiffTab.extended.site(meth.mat,sample.groups[[1]],sample.groups[[2]])
map.regions.to.sites <- regionMapping(rnb.set.example,"promoters")
dm.promoters <- computeDiffTab.default.region(dm.sites,map.regions.to.sites)
```

`computeDiffTab.default.site`

computeDiffTab.site

Description

computes a difference table containing multiple difference measures, In the simple version the difference in means, quotients in means and a p-value for the comparison of two groups in a table are computed. This is computed for each row of the input table. The extended version contains additional columns

Usage

```
computeDiffTab.default.site(
  X,
  inds.g1,
  inds.g2,
  diff.method = rnb.getOption("differential.site.test.method"),
  variability.method = rnb.getOption("differential.variability.method"),
  paired = FALSE,
  adjustment.table = NULL,
  eps = 0.01,
  imputed = FALSE
)

computeDiffTab.extended.site(
  X,
  inds.g1,
  inds.g2,
  diff.method = rnb.getOption("differential.site.test.method"),
  variability.method = rnb.getOption("differential.variability.method"),
  paired = FALSE,
  adjustment.table = NULL,
  eps = 0.01,
  covg = NULL,
  covg.thres = rnb.getOption("filtering.coverage.threshold"),
  imputed = FALSE
)
```

Arguments

`X`	Matrix on which the difference measures are calculated for every row
`inds.g1`	column indices of group 1 members
`inds.g2`	column indices of group 2 members
`diff.method`	Method to determine p-values for differential methylation. Currently supported are "ttest" for a two-sided Welch t-test, "refFreeEWAS" for adjusting for cell mixtures, and "limma" for p-values resulting from linear modeling of the transformed beta values (M-values) and using techniques from expression microarray analysis employed in the `limma` package.
`variability.method`	Method to determine p-values for differential variability. Currently supported are "diffVar" for the diffVar method implemented in the missMethyl bioconductor package, and "iEVORA".
`paired`	should a paired a analysis be performed. If `TRUE` then inds.g1 and inds.g2 should have exactly the same length and should be order, such that the first element of inds.g1 corresponds to the first element of inds.g2 and so on.
`adjustment.table`	a table of variables to be adjusted for in the differential methylation test. Currently this is only supported for `diff.method=="limma"`
`eps`	Epsilon for computing quotients (avoid division by 0 by adding this value to denominator and enumerator before calculating the quotient)
`imputed`	flag indicating if methylation matrix was already imputed

`covg`	coverage information (should be NULL for disabled or of equal dimensions as X)
`covg.thres`	a coverage threshold

Value

a dataframe containing the following variables:

`mean.g1`	Mean of group 1
`mean.g2`	Mean of group 2
`mean.diff`	Difference in means
`mean.quot.log2`	log2 of the quotient of means
`diffmeth.p.val`	P-value (as determined by `diff.method`)
`max.g1/max.g2`	[extended version only] Group maxima
`min.g1/min.g2`	[extended version only] Group minima
`sd.g1/sd.g2`	[extended version only] Group standard deviations
`min.diff`	[extended version only] Minimum of 0 and single linkage difference between the groups
`diffmeth.p.adj.fdr`	[extended version only] FDR adjusted p-values
`num.na.g1/num.na.g2`	[extended version only] number of NA methylation values for groups 1 and 2 respectively
`mean.covg.g1/mean.covg.g2`	[extended version with coverage information only] mean coverage of groups 1 and 2 respectively
`min.covg.g1/min.covg.g2`	[extended version with coverage information only] minimum coverage of groups 1 and 2 respectively
`max.covg.g1/max.covg.g2`	[extended version with coverage information only] maximum coverage of groups 1 and 2 respectively
`covg.thresh.nsamples.g1/2`	[extended version with coverage information only] number of samples in group 1 and 2 respectively exceeding the coverage threshold for this site.

Author(s)

Fabian Mueller

Examples

```
library(RnBeads.hg19)
data(small.example.object)
logger.start(fname=NA)
meth.mat <- meth(rnb.set.example)
sample.groups <- rnb.sample.groups(rnb.set.example)[[1]]
dm <- computeDiffTab.extended.site(meth.mat,sample.groups[[1]],sample.groups[[2]])
summary(dm)
```

`covg,RnBSet-method` *covg-methods*

Description

Extract coverage information from an object of `RnBSet` class.

Usage

```
## S4 method for signature 'RnBSet'
covg(object, type = "sites", row.names = FALSE, i = NULL, j = NULL)
```

Arguments

`object`	Dataset of interest.
`type`	`character` singleton. If `sites` DNA methylation information per each available site is returned. Otherwise should be one of region types for for which the summarized coverage information is available
`row.names`	Flag indicating of row names are to be generated in the result.
`i`	indices of sites/regions to be retrieved. By default (`NULL`), all will be retrieved.
`j`	indices of samples to be retrieved. By default (`NULL`), all will be retrieved.

Value

coverage information available for the dataset in the form of a `matrix`.

Examples

```
library(RnBeads.hg19)
data(small.example.object)
## per-site beta-value matrix
cvg<-covg(rnb.set.example, row.names=TRUE)
head(cvg)
```

`create.densityScatter` *create.densityScatter*

Description

Creates a density scatterplot highlighting points in sparsely populated plot regions as well as points marked as special in a seperate color

Usage

```
create.densityScatter(
  df2p,
  is.special = NULL,
  dens.subsample = FALSE,
  dens.special = TRUE,
  sparse.points = 0.01,
  dens.n = 100,
  add.text.cor = FALSE
)
```

Arguments

`df2p`	`data.frame` to be plotted. Only the fist two columns are taken into account as x and y coordinates respectively
`is.special`	boolean vector of length equal to the number of rows in `df2p`. Specifies which points should be highlighed seperately in a different color
`dens.subsample`	if the number of points exceeds this number, subsample the number of points for the density estimation to that number. Any non-numeric value disables subsampling.
`dens.special`	Flag indicating whether the points of the special population should be colored according to their density
`sparse.points`	Either percentage (`<=1,>=0`) or the absolute number of points in the sparsely populated area that should be drawn seperately. A value of 0 means that these points will not be drawn.
`dens.n`	passed on to `ggplot2::stat_density2d`: argument: `n`
`add.text.cor`	flag indicating whether a text token with the correlation coefficient should be included in the lower right corner of the plot

Value

`ggplot` object

Author(s)

Fabian Mueller

Examples

```
d <- data.frame(x=rnorm(1000),y=rnorm(1000))
s <- rep(FALSE,1000)
s[sample(1:length(s),100)] <- TRUE
create.densityScatter(d,s)
```

`create.hex.summary.plot` *create.hex.summary.plot*

Description

Creates a summary plot binning the data given by a certain quantity in heagonal bins

Usage

```
create.hex.summary.plot(
  df2p,
  x = colnames(df2p)[1],
  y = colnames(df2p)[2],
  q = colnames(df2p)[3],
  bins = 128,
  fun = median,
  ...
)
```

Arguments

`df2p`	`data.frame` to be plotted.
`x`	name of the variable in `df2p` considered as x-axis
`y`	name of the variable in `df2p` considered as y-axis
`q`	name of the variable in `df2p` considered as quantity to be summarized over bins
`bins`, `fun`, `...`	arguments to be passed on to `stat_summary_hex`

Value

`ggplot` object

Author(s)

Fabian Mueller

`create.scatter.dens.points` *create.scatter.dens.points*

Description

Creates a scatterplot containing all points in a given data.frame. Points are colored according to point density. Optionally, a selection of points are shown in a different color

Usage

```
create.scatter.dens.points(
  df2p,
  is.special = NULL,
  dens.special = TRUE,
  mock = FALSE
)
```

Arguments

`df2p`	`data.frame` to be plotted. Only the fist two columns are taken into account as x and y coordinates respectively
`is.special`	boolean vector of length equal to the number of rows in `df2p`. Specifies which points should be highlighed seperately in a different color
`dens.special`	Flag indicating whether the points of the special population should be colored according to their density
`mock`	Should only the axis be plotted? useful when exporting scatterplots with lots of points as immage and the corresponding axis as vector graphics.

Value

`ggplot` object

Author(s)

Fabian Mueller

Examples

```
d <- data.frame(x=rnorm(1000),y=rnorm(1000))
s <- rep(FALSE,1000)
s[sample(1:length(s),100)] <- TRUE
create.scatter.dens.points(d,s)
```

`createReport` *createReport*

Description

Creates a new report object.

Usage

```
createReport(
  fname,
  title,
  page.title = "RnBeads report",
  authors = NULL,
  dirs = NULL,
  init.configuration = FALSE
)
```

Arguments

`fname`	Single-element `character` vector denoting the name of the file to contain the HTML report. If this file already exists, it will be overwritten.
`title`	Title of the report in the form of a single-element `character` vector.
`page.title`	Web page title. This usually appears in the web browser's window title when the report is open. If specified, this must be a vector. Note that only the first element is used.
`authors`	Optional list of authors in the form of a `character` vector. This list is included in the header of the generated HTML file. Note that author names can contain only Latin leters, space, dash (`-`), comma (`,`) or dot (`.`).
`dirs`	Location of the supporting directories, that is, paths that are expected to contain additional files linked to from the HTML report. See the *Details* section for a list of these directories.
`init.configuration`	Flag indicating if the report configuration data should be initialized. If this parameter is `TRUE`, the method creates the respective directory and copies configuration files that define cascading style sheet (CSS) definitions and Javascript functions used by the HTML report. If such configuration files already exist, they will be overwritten. Since the aforementioned files can be shared by multiple reports, it is recommended that the configuration is initialized using the method `rnb.initialize.reports`, instead of setting this flag to `TRUE`.

Details

If specified, the parameter `dirs` must be a `character` vector. The following names are read:

- `"configuration"` Directory that contains the auxilliary configuration files, such as style sheets and Javascript files. If missing or `NA`, the default value used is `"configuration"`.
- `"data"` Directory to contain the tables, lists and other generated data files that are linked to in the HTML report. If missing or `NA`, the value used is formed from the file name `fname` (without the extension) and the suffix `"_data"`.
- `"pngs"` Directory to contain the low resolution PNG images shown in the HTML report. If missing or `NA`, the value used is formed from the file name `fname` (without the extension) and the suffix `"_images"`.
- `"pdfs"` Directory to contain the PDF images (if such are created). If not missing or `NA`, the value used is formed from the file name `fname` (without the extension) and the suffix `"_pdf"`.
- `"high"` Directory to contain the high resolution PNG images (if such are created). If missing or `NA`, the value used is the same as the `pngs` directory.

Any other elements, if present, are ignored. Note that these directories are not required to point to different locations. In particular, if the directories for low and for high resolution images are identical, the high-resolution image files are assumed to be the ones with suffix `"_high_resolution.png"`. See `createReportPlot` for creating image files. In order to ensure independence of the operating system, there are strong restrictions on the names of the file and directories. The name of the report's HTML file can consist of the following symbols only: Latin letters, digits, dot (`.`), dash (`-`) and underline (`_`). The extension of the report's HTML file must be one of `htm`, `html`, `xhtml` or `xml`. The supporting directories must be given as relative paths; the restrictions on the path names are identical to the ones for file name. Forward slash (`/`) is to be used as path separator. Path names cannot start or end with a slash. None of the directory names can be an empty string, use `"."` instead. A value in the form `"mypath/.html"` for `fname` is invalid. Upon initialization, the report attempts to create or overwrite the specified `fname`. If the path to it does not exist, or if the current process does not have permissions to write to the file, report initialization will fail. The report object visits each supporting directory (except `configuration`) and attempts to create it, unless it is an existing empty directory. Report initialization will fail if any of the visited directories does not meet the criteria and could not be created. Hidden files (file names starting with `"."` on Unix platforms) are ignored. Thus, all supporting directories that already exist and contain hidden files only are considered valid.

Value

Newly created `Report` object.

Author(s)

Yassen Assenov

See Also

`Report` for functions adding contents to an HTML report

Examples

```
report <- createReport("example.html", "Example", init.configuration = TRUE)
```

`createReportGgPlot` *createReportGgPlot*

Description

creates a report plot containing a `ggplot object`. Except for the ggp parameter, the signature and behavior is identical to `createReportPlot`.

Usage

```
createReportGgPlot(
  ggp,
  fname,
  report = NULL,
  width = 7,
  height = 7,
  create.pdf = TRUE,
  low.png = as.integer(100),
  high.png = as.integer(0)
)
```

Arguments

`ggp`	`ggplot` object to be plotted
`fname`	`character` vector with one element storing the name of the output file, without the extension. The initialized object appends `.pdf` and/or `.png` to this name.
`report`	Report (object of type `Report`) to which this plot is going to be added. This is used to set the directories for PDF and/or PNG files generated for these plots. If this parameter is `NULL`, the current working directory is used to host all generated images.
`width`	`numeric` storing the width of the device in inches. The length of this vector must be `1`.
`height`	`numeric` storing the height of the device in inches. The length of this vector must be `1`.
`create.pdf`	Flag indicating if a PDF image is to be created. The length of this vector must be `1`.
`low.png`	Resolution, in dots per inch, used for the figure image. Set this to `0` or a negative value to disable the creation of a low resolution image. The length of this vector must be `1`.
`high.png`	Resolution, in dots per inch, used for a dedicated image. Set this to `0` or a negative value to disable the creation of a high resolution image. The length of this vector must be `1`.

Value

Newly created `ReportGgPlot` object.

Author(s)

Fabian Mueller

`createReportPlot` *createReportPlot*

Description

Initializes a report plot and opens a device to create it. The type of the device created depends on the parameters `create.pdf`, `low.png` and `high.png`. If `create.pdf` is `TRUE`, a PDF device is opened and its contents are later copied to PNG device(s) if needed. Otherwise, a PNG device is opened. Note that at least one of the following conditions must be met:

- `create.pdf` == `TRUE`
- `low.png` > `0`
- `high.png` > `0`

Usage

```
createReportPlot(
  fname,
  report = NULL,
  width = 7,
  height = 7,
  create.pdf = TRUE,
  low.png = 100L,
  high.png = 0L
)
```

Arguments

`fname`	`character` vector with one element storing the name of the output file, without the extension. The initialized object appends `.pdf` and/or `.png` to this name.
`report`	Report (object of type `Report`) to which this plot is going to be added. This is used to set the directories for PDF and/or PNG files generated for these plots. If this parameter is `NULL`, the current working directory is used to host all generated images.
`width`	`numeric` storing the width of the device in inches. The length of this vector must be `1`.
`height`	`numeric` storing the height of the device in inches. The length of this vector must be `1`.
`create.pdf`	Flag indicating if a PDF image is to be created. The length of this vector must be `1`.
`low.png`	Resolution, in dots per inch, used for the figure image. Set this to `0` or a negative value to disable the creation of a low resolution image. The length of this vector must be `1`.
`high.png`	Resolution, in dots per inch, used for a dedicated image. Set this to `0` or a negative value to disable the creation of a high resolution image. The length of this vector must be `1`.

Details

In order to ensure independence of the operating system, there are strong restrictions on the name of the file. It can consist of the following symbols only: Latin letters, digits, dot (`.`), dash (`-`) and underline (`_`). The name must not include paths, that is, slash (`/`) or backslash (`\`) cannot be used.

Value

Newly created `ReportPlot` object.

Author(s)

Yassen Assenov

See Also

`pdf` for manually initializing a graphics device; `Report` for other functions adding contents to an HTML report

Examples

```
plot.image <- createReportPlot('scatterplot_tumors')
plot(x = c(0.4, 1), y = c(9, 3), type = 'p', main = NA, xlab = expression(beta), ylab = 'Measure')
off(plot.image)
```

`data.frame2GRanges` *data.frame2GRanges*

Description

Converts a `data.frame` that defines genomic regions to object of type `GRanges`.

Usage

```
data.frame2GRanges(
  dframe,
  ids = rownames(dframe),
  chrom.column = "Chromosome",
  start.column = "Start",
  end.column = "End",
  strand.column = NULL,
  assembly = "hg19",
  sort.result = TRUE
)
```

Arguments

`dframe`	Table defining genomic regions.
`ids`	Region names (identifiers) as a `character` vector, or `NULL` if no names are present.
`chrom.column`	Column name or index that lists the chromosome names.
`start.column`	Column name or index that lists the start positions of the regions.
`end.column`	Column name or index that lists the end positions of the regions.
`strand.column`	Column name or index that lists the strands on which the regions are located. Set this to `NULL` if this region set is not strand-specific.
`assembly`	Genome assembly of interest. See `rnb.get.assemblies` for the list of supported genomes.
`sort.result`	Should the resulting table be sorted

Value

`GRanges` object encapsulating all well defined regions on supported chromosomes, contained in `dframe`. Columns other that the ones listed as parameters in this function are included as metadata.

Author(s)

Yassen Assenov

`densRanks` *densRanks*

Description

Rank the points accordind to density of the region they fall in. Densities are computed as Kernel Density estimates. The method and parameters are implemented in analogy to `grDevices::densCols`

Usage

```
densRanks(x, y = NULL, nbin = 128, bandwidth)
```

Arguments

`x`	x-coordinate
`y`	y-coordinate
`nbin`	number of bins
`bandwidth`	bandwidth

Author(s)

Fabian Mueller

`destroy,RnBDiffMeth-method`
destroy-methods

Description

remove tables stored to disk from the file system. Useful for cleaning up disk dumped objects. CAUTION: currently only works with reloaded objects

Usage

```
## S4 method for signature 'RnBDiffMeth'
destroy(object)
```

Arguments

`object`	`RnBDiffMeth` object

Value

Nothing of particular interest

Author(s)

Fabian Mueller

`destroy,RnBSet-method` *destroy-methods*

Description

Remove tables stored to disk from the file system. Useful for cleaning up disk dumped objects.

Usage

```
## S4 method for signature 'RnBSet'
destroy(object)

## S4 method for signature 'RnBeadSet'
destroy(object)

## S4 method for signature 'RnBeadRawSet'
destroy(object)
```

Arguments

`object`	object inheriting from `RnBSet`

Value

Nothing of particular interest

`deviation.plot.beta` *deviation.plot.beta*

Description

Creates a deviation plot based on the methylation beta values of a population.

Usage

```
deviation.plot.beta(betas, c.values = NULL, c.legend = NULL)
```

Arguments

`betas`	Non-empty numeric `matrix` of methylation beta values. Rows in this matrix must denote sites or regions, and columns - samples. If a locus (row in the matrix) contains missing values only, it is not included in the plot.
`c.values`	Vector (usually a `factor`) storing category or quantitative values for each site or region. The length of this vector must be equal to `nrow(betas)`, the *i*-th element storing the property values for the *i*-th locus in `betas`. Note that this vector's names, if present, are ignored.

`c.legend`	If `c.values` stores categories, this parameter specifies the mapping from property values to colors. The mapping is in the form of a named `character` vector. All values that appear in `c.values` must be present among the names of this vector. The order of the values in this mapping determines in which order the colors are stacked (when the number of loci is large). If `c.values` denotes a quantitative measure, this parameter is a singleton `integer`, specifying the color scheme for visualizing the values. Currently, the only supported values are `2` and `3`. See `rnb.options` for more details.

Value

Methylation variability as a number between `0` and `1`, invisibly. This number denotes the relative area of variation in the generated plot.

Author(s)

Yassen Assenov

`diffVar` *diffVar*

Description

This routine applies the diffVar method from the `missMethyl` package that determines sites exhibiting differential variability between two sample groups

Usage

```
diffVar(meth.matrix, inds.g1, inds.g2, adjustment.table = NULL, paired = FALSE)
```

Arguments

`meth.matrix`	Matrix containing the methylation information used to calculate differentially variable sites between the two groups
`inds.g1`	Indices in the phenotypic table corresponding to the first group.
`inds.g2`	Indices in the phenotypic table corresponding to the second group.
`adjustment.table`	A `data.frame` containing variables to adjust for in the testing
`paired`	Should the analysis be performed in a paired fashion. If yes, the first index in `inds.g1` must correspond to the first in `inds.g2` and so on.

Value

P-values as the result of the diffVar method not adjusted for multiple hypothesis testing.

Author(s)

Michael Scherer

References

Phipson, Belinda, Oshlack, Alicia (2014) DiffVar: a new method for detecting differential variability with application to methylation in cancer and aging Genome Biology 15(9):465.

`dim,BigFfMat-method` *Dimensions of BigFfMat*

Description

Dimensions of BigFfMat

Usage

```
## S4 method for signature 'BigFfMat'
dim(x)
```

Arguments

`x`	BigFfMat object

`downloadLolaDbs` *downloadLolaDbs*

Description

Downloading prepared LOLA DBs from server

Usage

```
downloadLolaDbs(dest, dbs = c("LOLACore"))
```

Arguments

`dest`	destination directory
`dbs`	vector of names of LOLA DBs to be downloaded. Currently 'LOLACore' and 'LOLAExt' are supported

Details

Requires a stable internet connection. Could take a while depending on the size of the database and the internet connection

Value

a list containing vectors of directory names for each available genome assembly

Author(s)

Fabian Mueller

Examples

```
lolaDest <- tempfile()
dir.create(lolaDest)
lolaDirs <- downloadLolaDbs(lolaDest, dbs="LOLACore")
```

`dpval,RnBeadSet-method` *dpval-methods*

Description

Extract detection p-values from an object of `RnBeadSet` class.

Usage

```
## S4 method for signature 'RnBeadSet'
dpval(object, type = "sites", row.names = FALSE, i = NULL, j = NULL)
```

Arguments

`object`	`RnBeadSet` or `RnBeadRawSet` object
`type`	`character` singleton. If `sites` detection p-values per each available site is returned. Otherwise should be one of region types for for which the summarized p-values are available
`row.names`	Flag indicating of row names are to be generated in the result.
`i`	Indices of sites/regions to be retrieved. By default (`NULL`), all will be retrieved.
`j`	Indices of samples to be retrieved. By default (`NULL`), all will be retrieved.

Value

detection p-values available for the dataset in the form of a `matrix`.

Examples

```
library(RnBeads.hg19)
data(small.example.object)
dp<-dpval(rnb.set.example, row.names=TRUE)
head(dp)
```

`estimateProportionsCP` *estimateProportionsCP*

Description

Estimates cell type proportions using the constrained projection method from Houseman et al. [1]

Usage

```
estimateProportionsCP(
  rnb.set,
  cell.type.column,
  n.most.variable = NA,
  n.markers = 500L,
  constrained = TRUE,
  full.output = FALSE
)
```

Arguments

`rnb.set`	RnBSet object
`cell.type.column`	integer index or character identifier of a column in the RnBSet object sample annotation table which gives the mapping to reference cell type samples
`n.most.variable`	Singleton integer specifying how many top variable CpGs should be used for marker selection. If this option is set to `NA` or `NULL`, all sites are considered. Please take into account the extended computation time in such a case.
`n.markers`	singleton integer specifying how many CpGs should be used as markers for fitting the projection model
`constrained`	if `TRUE` the returned cell type proportion estimates are non-negative
`full.output`	if `TRUE` not only the estimated proportions but also the intermediate analysis results are returned

Details

This is a minimally customized implementation of the method by Houseman et al. [1] based on the orginial code kindly provided by Andres Houseman. Note that RnBeads does not provide any reference data sets, and the methylomes of purified cell types should be provided by the user as a part of the object supplied via `rnb.set`. The column specified by `cell.type.column` should give assignment of each reference methylome replicate to a cell type and missing values for all the target samples. First the marker selection model is fit to estimate association of each CpG with the given reference cell types (first expression in eq. (1) of [1]). The strength of association is expressed as an F-statistic. Since fitting the marker selection model to all CpGs can take a lot of time, one can limit the marker search only to variable CpG positions by setting `n.most.variable` to non-`NA` positive integer. The CpGs will be ranked using across-sample variance in the reference data set and `n.most.variable` will be taken to fit the marker selection model. Coefficients of the fit, together with the F-statistic value for each CpG, are returned in case `full.output` is `TRUE`. Thereafter, `n.markers` are selected as true quantitative markers and the projection model (eq. [2]) is fit to estimate contributions of each cell type. Depending on the value of `constrained` the returned coefficients can be either raw or enforced to attain values between 0 and 1 with within-sample sum less or equal to 1.

Value

a matrix of estimated cell type contributions (samples times cell types) or a list with results of the intermetidate steps (see details).

Note

Requires the package **nlme**.

Author(s)

Pavlo Lutsik

References

1. Houseman, Eugene and Accomando, William and Koestler, Devin and Christensen, Brock and Marsit, Carmen and Nelson, Heather and Wiencke, John and Kelsey, Karl. DNA methylation arrays as surrogate measures of cell mixture distribution. BMC Bioinformatics 2012, 13:86

`exportDMRs2regionFile` *exportDMRs2regionFile*

Description

export differentially methylated regions to region file (standard bed). The output is in BED6 format where the score corresponds to to the combined rank (rank==1 would receive a score of 1000 and a combined rank equal to the number of regions a score of 0)

Usage

```
exportDMRs2regionFile(
  rnbSet,
  diffmeth,
  dest,
  comp.name,
  region.type,
  rank.cut = NULL,
  rerank = FALSE
)
```

Arguments

`rnbSet`	the RnBSet object for which the DMRs were computed.
`diffmeth`	DiffMeth object. See `rnb.execute.computeDiffMeth` for details.
`dest`	destination file name
`comp.name`	name of the comparison
`region.type`	region type.
`rank.cut`	rank cutoff. If `NULL` (default), all regions are processed.
`rerank`	flag indicating whether the ranks should be reranked or whether `rank.cut` refers to the absolute rank

Value

`NULL`

Author(s)

Fabian Mueller

Examples

```
library(RnBeads.hg19)
data(small.example.object)
logger.start(fname=NA)
dm <- rnb.execute.computeDiffMeth(rnb.set.example,pheno.cols=c("Sample_Group","Treatment"))
exportDMRs2regionFile(rnb.set.example,dm,tempfile(),get.comparisons(dm)[1],"promoters")
```

`get.adjustment.variables` *get.adjustment.variables*

Description

Given indices for two groups of samples for comparison, this function retrieves `data.frame` containing the variables to be adjusted for

Usage

```
get.adjustment.variables(
  rnbSet,
  inds.g1,
  inds.g2 = -inds.g1,
  colnames.adj = c(),
  colname.target = "",
  adjust.sva = FALSE,
  adjust.celltype = FALSE
)
```

Arguments

`rnbSet`	RnBSet object
`inds.g1`	sample indices in `rnbSet` of group 1 members
`inds.g2`	sample indices in `rnbSet` of group 2 members
`colnames.adj`	column names in `pheno(rnbSet)` to retrieve
`colname.target`	column names in `pheno(rnbSet)` of the target variable. Only important if `adjust.sva==TRUE`
`adjust.sva`	flag indicating whether the resulting table should also contain surrogate variables (SVs) for the given target variable.
`adjust.celltype`	flag indicating whether the resulting table should also contain estimated celltype contributions. See `rnb.execute.ct.estimation` for details.

Value

a `data.frame` containing one column for each selected variable from the phenotypic data each row corresponds to a sample in the union of samples of the wto groups with the first `length(inds.g1)` rows corresponding to group 1 and the remaining rows corresponding to group 2

Author(s)

Fabian Mueller

Examples

```
library(RnBeads.hg19)
data(small.example.object)
logger.start(fname=NA)
sample.groups <- rnb.sample.groups(rnb.set.example)[[1]]
get.adjustment.variables(rnb.set.example,sample.groups[[1]],sample.groups[[2]],"Cell_Line")
```

`get.comparison.grouplabels,RnBDiffMeth-method` *get.comparison.grouplabels-methods*

Description

Gets all comparison grouplabels represented in the object as character matrix of dimension n.comparisons x 2 where the columns specify group names 1 and 2 respectively

Usage

```
## S4 method for signature 'RnBDiffMeth'
get.comparison.grouplabels(object)
```

Arguments

`object`	`RnBDiffMeth` object

Value

character matrix containing comparison group names

Author(s)

Fabian Mueller

Examples

```
library(RnBeads.hg19)
data(small.example.object)
logger.start(fname=NA)
dm <- rnb.execute.computeDiffMeth(rnb.set.example,pheno.cols=c("Sample_Group","Treatment"))
get.comparison.grouplabels(dm)
```

`get.comparison.groupsizes,RnBDiffMeth-method` *get.comparison.groupsizes-methods*

Description

Gets all comparison group sizes represented in the object as character matrix of dimension n.comparisons x 2 where the columns specify sizes of groups 1 and 2 respectively

Usage

```
## S4 method for signature 'RnBDiffMeth'
get.comparison.groupsizes(object)
```

Arguments

`object`	`RnBDiffMeth` object

Value

character matrix containing comparison group sizes

Author(s)

Fabian Mueller

Examples

```
library(RnBeads.hg19)
data(small.example.object)
logger.start(fname=NA)
dm <- rnb.execute.computeDiffMeth(rnb.set.example,pheno.cols=c("Sample_Group","Treatment"))
get.comparison.groupsizes(dm)
```

`get.comparison.info` *get.comparison.info*

Description

retrieve the comparison information for an RnBSet object

Usage

```
get.comparison.info(
  x,
  pheno.cols = rnb.getOption("differential.comparison.columns"),
  region.types = rnb.region.types.for.analysis(x),
 pheno.cols.all.pairwise = rnb.getOption("differential.comparison.columns.all.pairwise"),
  columns.pairs = rnb.getOption("columns.pairing"),
  columns.adj = rnb.getOption("covariate.adjustment.columns"),
  adjust.sva = rnb.getOption("differential.adjustment.sva"),
  pheno.cols.adjust.sva = rnb.getOption("inference.targets.sva"),
  adjust.celltype = rnb.getOption("differential.adjustment.celltype"),
  adjust.na.rm = TRUE
)
```

Arguments

`x`	`RnBSet` object
`pheno.cols`	column names of the pheno slot in `x` on which the dataset should be partitioned. Those columns are required to be factors or logical. In case of factors, each group in turn will be compared to all other groups
`region.types`	which region types should be processed for differential methylation
`pheno.cols.all.pairwise`	integer or character vector specifying the colomns of `pheno(x)` on which all pairwise comparisons should be conducted. A value of `NULL` indicates no columns.
`columns.pairs`	argument passed on to `rnb.sample.groups`. See its documentation for details.
`columns.adj`	Column names or indices in the table of phenotypic information to be used for confounder adjustment in the differential methylation analysis.
`adjust.sva`	flag indicating whether the adjustment table should also contain surrogate variables (SVs) for the given target variable.
`pheno.cols.adjust.sva`	Target variables for SVA adjustment. Only important if `adjust.sva==TRUE`. Only the intersection of `pheno.cols` and `pheno.cols.adjust.sva` is considered for SVA adjustment.
`adjust.celltype`	flag indicating whether the resulting table should also contain estimated celltype contributions. See `rnb.execute.ct.estimation` for details.
`adjust.na.rm`	Flag indicating whether NAs in the adjustment table should be removed.

Value

a list containing one element for each comparison to be conducted. Each element is again a list containing:

`comparison`	the name of the comparison
`pheno.colname`	the column name of the sample annotation table the comparison is derived from
`group.names`	the names of the two groups being compared
`group.inds`	the sample indices of the samples belonging to the two groups
`paired`	flag indicating whether paired analysis is conducted
`adj.sva`	flag indicating whether adjustment for SVA is conducted

`adj.celltype`	flag indicating whether adjustment for cell type is conducted
`adjustment.table`	the covariate adjustment table. `NULL` if the comparison is not adjusted
`region.types`	the region types applicable to the analysis

Author(s)

Fabian Mueller

Examples

```
library(RnBeads.hg19)
data(small.example.object)
logger.start(fname=NA)
cmp.info <- get.comparison.info(rnb.set.example,pheno.cols=c("Sample_Group","Treatment"))
cmp.info[[1]]
```

`get.comparisons,RnBDiffMeth-method`

get.comparisons-methods

Description

Gets all comparisons represented in the object as character vector

Usage

```
## S4 method for signature 'RnBDiffMeth'
get.comparisons(object)
```

Arguments

`object`	`RnBDiffMeth` object

Value

character vector containing comparisons

Author(s)

Fabian Mueller

Examples

```
library(RnBeads.hg19)
data(small.example.object)
logger.start(fname=NA)
dm <- rnb.execute.computeDiffMeth(rnb.set.example,pheno.cols=c("Sample_Group","Treatment"))
get.comparisons(dm)
```

`get.covariates.ct` *get.covariates.ct*

Description

Retrieves an NxK matrix of cell type contributions stored in an RnBSet for a given target variable

Usage

```
get.covariates.ct(rnb.set)
```

Arguments

`rnb.set`	`RnBSet` object

Value

an NxK matrix of K cell types contributions for N samples of the `rnb.set`. `NULL` if the components have not been computed or added to `rnb.set`.

`get.covariates.sva` *get.covariates.sva*

Description

Retrieves an NxK table of Surrogate variables stored in an RnBSet for a given target variable

Usage

```
get.covariates.sva(rnb.set, target)
```

Arguments

`rnb.set`	`RnBSet` object
`target`	target variable. Must be in `pheno(rnb.set)` and belong to target variables for which the SVs have already been computed and stored in the RnBSet.

Value

an NxK table of K Surrogate variables stored for N samples of the `rnb.set`. `NULL` if the components have not been computed or added to `rnb.set`.

Author(s)

Fabian Mueller

Examples

```
library(RnBeads.hg19)
data(small.example.object)
logger.start(fname=NA)
sva.obj <- rnb.execute.sva(rnb.set.example,c("Sample_Group","Treatment"),numSVmethod="be")
sva.obj$sva.performed
sva.obj$num.components
rnb.set.mod <- set.covariates.sva(rnb.set.example, sva.obj)
get.covariates.sva(rnb.set.mod,"Sample_Group")
```

`get.covg.thres,RnBDiffMeth-method`

get.covg.thres-methods

Description

Gets the coverage threshold employed for obtaining statistics in the differential methylation tables

Usage

```
## S4 method for signature 'RnBDiffMeth'
get.covg.thres(object)
```

Arguments

`object`	RnBDiffMeth object

Value

integer coverage threshold

Author(s)

Fabian Mueller

Examples

```
library(RnBeads.hg19)
data(small.example.object)
logger.start(fname=NA)
dm <- rnb.execute.computeDiffMeth(rnb.set.example,pheno.cols=c("Sample_Group","Treatment"))
get.covg.thres(dm)
```

`get.cpg.stats` *get.cpg.stats*

Description

Computes CpG-related statistics for the specified regions.

Usage

```
get.cpg.stats(chrom.sequence, starts, ends)
```

Arguments

`chrom.sequence`	Chromosome sequence, usually obtained from the assembly's genome definition. This must be an object of type `MaskedDNAString`.
`starts`	`integer` vector of start positions for the regions of interest.
`ends`	`integer` vector of end positions for the regions of interest.

Value

Table of statistics for the regions in the form of a `matrix` with the following columns: `"CpG"` and `"GC"`. The columns contain the number of CpG dinucleoties and the number of C and G bases in each region.

Author(s)

Yassen Assenov

`get.files` *get.files*

Description

Gets the list of all files that are planned to be generated, or were already generated by the given report plot.

Usage

```
get.files(report.plot)
```

Arguments

`report.plot`	Report plot of interest. This must be an object of type `ReportPlot`.

Value

Non-empty `character` vector of absolute file names.

Author(s)

Yassen Assenov

Examples

```
plot.image <- createReportPlot('scatterplot', high.png = 200)
get.files(plot.image)
```

`get.region.types,RnBDiffMeth-method`

get.region.types-methods

Description

Gets all region types represented in the object as character vector

Usage

```
## S4 method for signature 'RnBDiffMeth'
get.region.types(object)
```

Arguments

`object`	`RnBDiffMeth` object

Value

character vector containing region types

Author(s)

Fabian Mueller

Examples

```
library(RnBeads.hg19)
data(small.example.object)
logger.start(fname=NA)
dm <- rnb.execute.computeDiffMeth(rnb.set.example,pheno.cols=c("Sample_Group","Treatment"))
get.region.types(dm)
```

---

get.site.test.method,RnBDiffMeth-method
get.site.test.method-methods

Description

Gets the site testing method used to obtain the p-values in the differential methylation tables

Usage

```
## S4 method for signature 'RnBDiffMeth'
get.site.test.method(object)
```

Arguments

`object`	RnBDiffMeth object

Value

character describing the site test method

Author(s)

Fabian Mueller

Examples

```
library(RnBeads.hg19)
data(small.example.object)
logger.start(fname=NA)
dm <- rnb.execute.computeDiffMeth(rnb.set.example,pheno.cols=c("Sample_Group","Treatment"))
get.site.test.method(dm)
```

---

get.table,RnBDiffMeth-method
get.table-methods

Description

Gets a differential methylation table

Usage

```
## S4 method for signature 'RnBDiffMeth'
get.table(
  object,
  comparison,
  region.type,
  undump = TRUE,
  return.data.frame = FALSE
)
```

Arguments

`object`	`RnBDiffMeth` object
`comparison`	character or index of the comparison of the table to retrieve
`region.type`	character or index of the region type of the table to retrieve
`undump`	Flag indicating whether to convert the table into a matrix instead of using the file descriptor. Only meaningful if the if the objects's `disk.dump` slot is true.
`return.data.frame`	should a data.frame be returned instead of a matrix?

Value

differential methylation table. See `computeDiffMeth.bin.site` and `computeDiffMeth.bin.region` for details.

Author(s)

Fabian Mueller

Examples

```
library(RnBeads.hg19)
data(small.example.object)
logger.start(fname=NA)
dm <- rnb.execute.computeDiffMeth(rnb.set.example,pheno.cols=c("Sample_Group","Treatment"))
dm.promoters <- get.table(dm,get.comparisons(dm)[1],"promoters",return.data.frame=TRUE)
summary(dm.promoters)
```

`get.table.ids` *Returns the colum names of the differential variability table.*

Description

Returns the colum names of the differential variability table.

Usage

```
get.table.ids(includeCovg = FALSE)
```

Arguments

`includeCovg`	Flag indicating if dataset contains coverage information

Value

Column names of the differential variability table

`get.variability.method,RnBDiffMeth-method` *get.variability.method-methods*

Description

Gets the variability testing method used to obtain the p-values in the differential varibiality tables

Usage

```
## S4 method for signature 'RnBDiffMeth'
get.variability.method(object)
```

Arguments

`object`	RnBDiffMeth object

Value

character describing the variability method

Author(s)

Michael Scherer

`getCellTypesFromLolaDb` *getCellTypesFromLolaDb*

Description

retrieve or guess cell types from a LOLA DB object

Usage

```
getCellTypesFromLolaDb(lolaDb)
```

Arguments

`lolaDb`	LOLA DB object as returned by `LOLA::loadRegionDB` or `loadLolaDbs`

Value

character vector with cell types

Author(s)

Fabian Mueller

Examples

```
# download LOLA DB
lolaDest <- tempfile()
dir.create(lolaDest)
lolaDirs <- downloadLolaDbs(lolaDest, dbs="LOLACore")
lolaDb <- loadLolaDbs(lolaDirs[["hg19"]])
getCellTypesFromLolaDb(lolaDb)
```

`getExecutable,ClusterArchitecture,character-method` *getExecutable-methods*

Description

Retrieves the executable associated with a name/identifier

Usage

```
## S4 method for signature 'ClusterArchitecture,character'
getExecutable(object, exec.name)
```

Arguments

`object`	`ClusterArchitecture` object
`exec.name`	The executable's name/identifier

Value

The executable. If the name is not associated with any executable, the names will be returned and a warning will be raised

Author(s)

Fabian Mueller

`getModuleNumCores,RnBClusterRun-method` *getModuleNumCores-methods*

Description

Retrieves the number of cores used by each module

Usage

```
## S4 method for signature 'RnBClusterRun'
getModuleNumCores(object)
```

Arguments

`object`	`RnBClusterRun` object

Value

A named vector containing the number of cores for each module

Author(s)

Fabian Mueller

`getNamesFromLolaDb` *getNamesFromLolaDb*

Description

get human readable names from a LOLA DB object

Usage

```
getNamesFromLolaDb(lolaDb, addCollectionNames = FALSE, addDbId = TRUE)
```

Arguments

`lolaDb`	LOLA DB object as returned by `LOLA::loadRegionDB` or `loadLolaDbs`
`addCollectionNames`	attach the name of the collection to the name
`addDbId`	attach the index of the item in the LOLA DB object to the name

Value

character vector with human readable names

Author(s)

Fabian Mueller

Examples

```
# download LOLA DB
lolaDest <- tempfile()
dir.create(lolaDest)
lolaDirs <- downloadLolaDbs(lolaDest, dbs="LOLACore")
lolaDb <- loadLolaDbs(lolaDirs[["hg19"]])
getNamesFromLolaDb(lolaDb)
```

---

getNumNaMeth,RnBSet-method

getNumNaMeth-methods

Description

for each site/region, the getNumNaMeth retrieves the number of NA values accross all samples. Does this efficiently by breaking down the methylation matrix into submatrices

Usage

```
## S4 method for signature 'RnBSet'
getNumNaMeth(object, type = "sites", chunkSize = 1e+05, mask = NULL)
```

Arguments

`object`	object inheriting from `RnBSet`
`type`	"sites" or region type
`chunkSize`	size of each submatrix (performance tuning parameter)
`mask`	logical matrix. its entries will also be considered NAs in counting

Value

vector containing the number of NAs per site/region

---

getSubCmdStr,ClusterArchitecture-method

getSubCmdStr-methods

Description

Returns a string for the of command line corresponding to submitting a job with the given command to the cluster.

Usage

```
## S4 method for signature 'ClusterArchitecture'
getSubCmdStr(object, ...)
```

Arguments

`object`	`ClusterArchitecture` object
`...`	arguments passed on to `getSubCmdTokens,ClusterArchitecture-method`

Value

A string containing the submission command

Author(s)

Fabian Mueller

`getSubCmdTokens,ClusterArchitecture-method`

getSubCmdTokens-methods

Description

Returns a string for the of command line corresponding to submitting a job with the given command to the cluster.

Usage

```
## S4 method for signature 'ClusterArchitecture'
getSubCmdTokens(
  object,
  cmd.tokens,
  log,
  job.name = "",
  res.req = character(0),
  depend.jobs = character(0)
)
```

Arguments

`object`	`ClusterArchitecture` object
`cmd.tokens`	a character vector specifying the executable command that should be wrapped in the cluster submission command
`log`	file name and path of the log file that the submitted job writes to
`job.name`	name of the submitted job
`res.req`	character vector specifying required resources. The resource requirements should be the values of the vector, the names should specify the resource name
`depend.jobs`	character vector containg names or ids of jobs the submitted job will depend on.

Details

For a concrete child class implementation for a sun grid architecture specification see `getSubCmdTokens,ClusterArchit`

Value

A character vector containing the submission command tokens

Author(s)

Fabian Mueller

`getSubCmdTokens,ClusterArchitectureLSF-method`

getSubCmdTokens-methods

Description

Returns a string for the of command line corresponding to submitting a job with the given command to the cluster.

Usage

```
## S4 method for signature 'ClusterArchitectureLSF'
getSubCmdTokens(
  object,
  cmd.tokens,
  log,
  job.name = "",
  res.req = character(0),
  depend.jobs = character(0)
)
```

Arguments

`object`	`ClusterArchitectureLSF` object
`cmd.tokens`	a character vector specifying the executable command that should be wrapped in the cluster submission command
`log`	file name and path of the log file that the submitted job writes to
`job.name`	name of the submitted job
`res.req`	named vector of requested resources. Two options are available: `"clock.limit"` and `"memory.size"`
`depend.jobs`	character vector containg names or ids of jobs the submitted job will depend on.

Details

For a concrete child class implementation for a LSF architecture specification see `ClusterArchitectureLSF`

Value

A character vector containing the submission command tokens

Author(s)

Michael Scherer

Examples

```
arch <- new("ClusterArchitectureLSF",
 name="my_lsf_architecture"
)
getSubCmdTokens(arch,c("Rscript","my_great_script.R"),"my_logfile.log")
```

`getSubCmdTokens,ClusterArchitectureSGE-method` *getSubCmdTokens-methods*

Description

Returns a string for the of command line corresponding to submitting a job with the given command to the cluster.

Usage

```
## S4 method for signature 'ClusterArchitectureSGE'
getSubCmdTokens(
  object,
  cmd.tokens,
  log,
  job.name = "",
  res.req = character(0),
  depend.jobs = character(0),
  sub.binary = TRUE,
  quote.cmd = TRUE,
  queue = NULL
)
```

Arguments

`object`	`ClusterArchitectureSGE` object
`cmd.tokens`	a character vector specifying the executable command that should be wrapped in the cluster submission command
`log`	file name and path of the log file that the submitted job writes to
`job.name`	name of the submitted job
`res.req`	character vector specifying required resources. The resource requirements should be the values of the vector, the names should specify the resource name
`depend.jobs`	character vector containg names or ids of jobs the submitted job will depend on.
`sub.binary`	treat the command as binary (see `-b` flag of `qsub` of the SGE documentation)
`quote.cmd`	Flag indicating whether the submitted cammed should also be wrapped in quotes
`queue`	The name of the queue to submit jobs to

Details

For a concrete child class implementation for a sun grid architecture specification see `ClusterArchitectureSGE`

Value

A character vector containing the submission command tokens

Author(s)

Fabian Mueller

Examples

```
arch <- new("ClusterArchitectureSGE",
 name="my_sge_architecture"
)
getSubCmdTokens(arch,c("Rscript","my_great_script.R"),"my_logfile.log")
```

---

getSubCmdTokens,ClusterArchitectureSLURM-method

getSubCmdTokens-methods

Description

Returns a string for the of command line corresponding to submitting a job with the given command to the cluster.

Usage

```
## S4 method for signature 'ClusterArchitectureSLURM'
getSubCmdTokens(
  object,
  cmd.tokens,
  log,
  job.name = "",
  res.req = character(0),
  depend.jobs = character(0),
  sub.binary = TRUE,
  quote.cmd = TRUE
)
```

Arguments

`object`	`ClusterArchitectureSLURM` object
`cmd.tokens`	a character vector specifying the executable command that should be wrapped in the cluster submission command
`log`	file name and path of the log file that the submitted job writes to
`job.name`	name of the submitted job
`res.req`	named vector of requested resources. Two options are available: `"clock.limit"` and `"memory.size"`
`depend.jobs`	character vector containg names or ids of jobs the submitted job will depend on.
`sub.binary`	flag indicating if the command is to be submitted using the `"wrap"` option of SLURM
`quote.cmd`	Flag indicating whether the submitted cammed should also be wrapped in quotes

Details

For a concrete child class implementation for a SLURM architecture specification see `ClusterArchitectureSLURM`

Value

A character vector containing the submission command tokens

Author(s)

Michael Scherer

Examples

```
arch <- new("ClusterArchitectureSLURM",
 name="my_slurm_architecture"
)
getSubCmdTokens(arch,c("Rscript","my_great_script.R"),"my_logfile.log")
```

`getTargetFromLolaDb` *getTargetFromLolaDb*

Description

retrieve or guess the target from a LOLA DB object. Here, target typically refers to antibodies for ChIP-seq experiments, but could also refer to other annotations (e.g. motifs in TF motif databases, annotation according to UCSC features etc.)

Usage

```
getTargetFromLolaDb(lolaDb)
```

Arguments

`lolaDb`	LOLA DB object as returned by `LOLA::loadRegionDB` or `loadLolaDbs`

Value

character vector with targets

Author(s)

Fabian Mueller

Examples

```
# download LOLA DB
lolaDest <- tempfile()
dir.create(lolaDest)
lolaDirs <- downloadLolaDbs(lolaDest, dbs="LOLACore")
lolaDb <- loadLolaDbs(lolaDirs[["hg19"]])
getTargetFromLolaDb(lolaDb)
```

`greedycut.filter.matrix`

greedycut.filter.matrix

Description

Performs all iterations of the Greedycut algorithm for removing rows and columns from the given matrix.

Usage

```
greedycut.filter.matrix(mm, rows2ignore = integer(), rc.ties = "row")
```

Arguments

`mm`	Numeric matrix to filter.
`rows2ignore`	`integer` vector containing indices of rows in `mm` to be ignored by this function.
`rc.ties`	Flag indicating what the behaviour of the algorithm should be in case of ties between values of rows and columns. The value of this parameter must be one of `"row"`, `"column"` or `"any"` (the last one indicating random choice).

Value

Table summarizing the iterations of the algorithm in the form of a `data.frame` with the following columns : Index, Type, Score, Normalized score, Rows, Columns.

Author(s)

Yassen Assenov

See Also

`greedycut.get.submatrix` for extracting the resulting matrix after filtering

`greedycut.get.statistics`

greedycut.get.statistics

Description

Calculates various statistics on the iterations of Greedycut.

Usage

```
greedycut.get.statistics(filterinfo)
```

Arguments

`filterinfo`	Information on the filtering iterations as a `data.frame` returned by `greedycut.filter.matrix`.

Value

Additional statistics on the iterations in the form of a `data.frame` with the following columns: `"Elements retained"`, `"Elements removed"`, `"Mismatches retained"`, `"Mismatches removed"`, `"False Positive Rate"`, `"Sensitivity"`, `"D"`. The last column signifies distance from the diagonal in a ROC curve.

Author(s)

Yassen Assenov

`greedycut.get.submatrix`

greedycut.get.submatrix

Description

Filters a data matrix executing the given number of iterations of Greedycut.

Usage

```
greedycut.get.submatrix(
  mm,
  filter.info,
  it.num = nrow(filter.info) - as.integer(1)
)
```

Arguments

`mm`	Data `matrix` to be filtered.
`filter.info`	Information on the filtering iterations as a `data.frame` returned by `greedycut.filter.matrix`.
`it.num`	Number of iterations to execute. Defaults to all iterations.

Value

Data matrix containing subsets of the rows and columns of `mm`.

Author(s)

Yassen Assenov

`has.covariates.ct` *has.covariates.ct*

Description

Checks whether the given `RnBSet` object contains cell type contribution estimates

Usage

```
has.covariates.ct(rnb.set)
```

Arguments

`rnb.set`	`RnBSet` object

Value

`TRUE` if the supplied object contains the cell type covariates information and `FALSE` otherwise

---

`has.covariates.sva` *has.covariates.sva*

Description

Returns whether Surrogate Variables have been computed and added to the `rnb.set` for a given target variable

Usage

```
has.covariates.sva(rnb.set, target)
```

Arguments

`rnb.set`	`RnBSet` object
`target`	target variable. Must be in `pheno(rnb.set)` and belong to target variables for which the SVs have already been computed and stored in the RnBSet.

Value

`logical(1)`

Author(s)

Fabian Mueller

Examples

```
library(RnBeads.hg19)
data(small.example.object)
logger.start(fname=NA)
sva.obj <- rnb.execute.sva(rnb.set.example,c("Sample_Group","Treatment"),numSVmethod="be")
sva.obj$sva.performed
sva.obj$num.components
rnb.set.mod <- set.covariates.sva(rnb.set.example, sva.obj)
has.covariates.sva(rnb.set.example,"Sample_Group")
has.covariates.sva(rnb.set.mod,"Sample_Group")
has.covariates.sva(rnb.set.mod,"Treatment")
```

`hasCovg,RnBSet-method` *hasCovg-methods*

Description

Returns `TRUE` if the `RnBSet` object contains coverage information for sites or the specified region type.

Usage

```
## S4 method for signature 'RnBSet'
hasCovg(object, type = "sites")
```

Arguments

`object`	`RnBSet` of interest.
`type`	`character` singleton. If `sites` or a region type summarized in the object

Value

`TRUE` if the `RnBSet` object contains coverage information for sites or the specified region type. `FALSE` otherwise

Examples

```
library(RnBeads.hg19)
data(small.example.object)
## per-site beta-value matrix
hasCovg(rnb.set.example)
```

`includes.sites,RnBDiffMeth-method` *includes.sites-methods*

Description

Returns `TRUE` if the differential methylation object contains site-level information

Usage

```
## S4 method for signature 'RnBDiffMeth'
includes.sites(object)
```

Arguments

`object`	RnBDiffMeth object

Value

`TRUE` if the differential methylation object contains site-level information. `FALSE` otherwise

Author(s)

Fabian Mueller

Examples

```
library(RnBeads.hg19)
data(small.example.object)
logger.start(fname=NA)
dm <- rnb.execute.computeDiffMeth(rnb.set.example,pheno.cols=c("Sample_Group","Treatment"))
includes.sites(dm)
```

`initialize,ClusterArchitecture-method` *initialize.ClusterArchitecture*

Description

Initialize an ClusterArchitecture object

Usage

```
## S4 method for signature 'ClusterArchitecture'
initialize(.Object, name = "ClusterArchitecture")
```

Arguments

`.Object`	New instance of `ClusterArchitecture`.
`name`	A name or identifier

Author(s)

Fabian Mueller

`initialize,ClusterArchitectureLSF-method` *initialize.ClusterArchitectureLSF*

Description

Initialize an ClusterArchitecture object for a LSF

Usage

```
## S4 method for signature 'ClusterArchitectureLSF'
initialize(.Object, name = "ClusterArchitectureLSF", ...)
```

Arguments

`.Object`	New instance of `ClusterArchitectureLSF`.
`name`	A name or identifier
`...`	arguments passed on to the constructor of `ClusterArchitecture` (the parent class)

Author(s)

Michael Scherer

`initialize,ClusterArchitectureSGE-method` *initialize.ClusterArchitectureSGE*

Description

Initialize an ClusterArchitecture object for a Sun Grid Engine (SGE)

Usage

```
## S4 method for signature 'ClusterArchitectureSGE'
initialize(.Object, name = "ClusterArchitectureSGE", ...)
```

Arguments

`.Object`	New instance of `ClusterArchitectureSGE`.
`name`	A name or identifier
`...`	arguments passed on to the constructor of `ClusterArchitecture` (the parent class)

Author(s)

Fabian Mueller

`initialize,ClusterArchitectureSLURM-method`
initialize.ClusterArchitectureSLURM

Description

Initialize an ClusterArchitecture object for a SLURM

Usage

```
## S4 method for signature 'ClusterArchitectureSLURM'
initialize(.Object, name = "ClusterArchitectureSLURM", ...)
```

Arguments

`.Object`	New instance of `ClusterArchitectureSLURM`.
`name`	A name or identifier
`...`	arguments passed on to the constructor of `ClusterArchitecture` (the parent class)

Author(s)

Michael Scherer

`initialize,RnBClusterRun-method`
initialize.RnBClusterRun

Description

Initialize an RnBClusterRun object

Usage

```
## S4 method for signature 'RnBClusterRun'
initialize(.Object, architecture)
```

Arguments

`.Object`	New instance of `RnBClusterRun`.
`architecture`	A `ClusterArchitecture` object managing the settings for a scientific compute cluster.

Author(s)

Fabian Mueller

`initialize,RnBDiffMeth-method` *initialize.RnBDiffMeth*

Description

Initialize an RnBDiffMeth object

Usage

```
## S4 method for signature 'RnBDiffMeth'
initialize(
  .Object,
  site.test.method = rnb.getOption("differential.site.test.method"),
  variability.method = rnb.getOption("differential.variability.method"),
  covg.thres = rnb.getOption("filtering.coverage.threshold"),
  disk.dump = FALSE,
  disk.path = NULL
)
```

Arguments

`.Object`	New instance of `RnBDiffMeth`.
`site.test.method`	method which was applied to obtain the site-level p-values.
`variability.method`	method to be used to calculate differentially variable sites. Has to be one of: ‘diffVar’ or ‘iEVORA’.
`covg.thres`	coverage threshold. Important for certain columns of the differential methylation tables. See `computeDiffMeth.bin.site` and `computeDiffMeth.bin.region` for details.
`disk.dump`	Flag indicating whether the tables should be stored on disk rather than in the main memory
`disk.path`	Path on the disk for DMTs.Only meaningful if `disk.dump` is `TRUE`

Author(s)

Fabian Mueller

`intensities.by.color` *intensities.by.color*

Description

Rearranges information from "M" and "U" slots of a RnBeadsRawSet object by color channel.

Usage

```
intensities.by.color(
  raw.set,
  address.rownames = TRUE,
  add.oob = all(!is.null(M0(raw.set)), !is.null(U0(raw.set))),
  add.controls = !is.null(qc(raw.set)),
  add.missing = TRUE,
  re.separate = FALSE
)
```

Arguments

`raw.set`	Methylation dataset as an instance of `RnBeadRawSet` object.
`address.rownames`	if `TRUE` the rows of the returned matrices are named with the with the corresponding Illumina probe addresses
`add.oob`	if `TRUE` the "out-of-band" intensities are included
`add.controls`	if `TRUE` the control probe intensities are included
`add.missing`	if `TRUE` the rows for the probes missing in `raw.set` is imputed with `NA` values
`re.separate`	if `TRUE` the type I and type II intensities, as well as the out-of-band and control probe intensities (if set to `TRUE`), will be returned as separate elements per channel and not as concatenated rows.

Value

A `list` with elements `Cy3` and `Cy5` containing average bead intensities measured for each each probe in the green and red channels, respectively. Exception, if `re.separate` is `TRUE` a `list` with elements `Cy3.I`, `Cy5.I`, and `II` will be returned. The elements `Cy3.I.oob`, `Cy5.I.oob` and also `Cy3.ctl`, `Cy5.ctl` will be returned if the respective parameters (`add.oob` and `add.ctl`) are set to true.

Author(s)

Pavlo Lutsik, Nathan Steenbuck

`is.valid,RnBDiffMeth-method`
is.valid-methods

Description

Validate an RnBDiffMeth object, ie. verify that all differential methylation tables are specified and accounted for

Usage

```
## S4 method for signature 'RnBDiffMeth'
is.valid(object, verbose = FALSE)
```

Arguments

`object`	`RnBDiffMeth` object
`verbose`	print more info to the logger

Value

TRUE iff all differential methylation tables are present and accounted for

Author(s)

Fabian Mueller

Examples

```
library(RnBeads.hg19)
data(small.example.object)
logger.start(fname=NA)
dm1 <- rnb.execute.computeDiffMeth(rnb.set.example,"Sample_Group",c("genes","tiling"))
dm2 <- rnb.execute.computeDiffMeth(rnb.set.example,c("Sample_Group","Treatment"),"promoters")
dm.join1 <- join.diffMeth(dm1,dm2)
#The following joint object is invalid due to missing region type - comparison combinations
is.valid(dm.join1)
dm3 <- rnb.execute.computeDiffMeth(rnb.set.example,c("Treatment"),c("genes","tiling"))
dm.join2 <- join.diffMeth(dm.join1,dm3)
#After joining the missing information, the new object is valid
is.valid(dm.join2)
```

`isImputed,RnBSet-method`

isImputed

Description

Getter for the imputation field. Return TRUE, if the object has been imputed and FALSE otherwise.

Usage

```
## S4 method for signature 'RnBSet'
isImputed(object)
```

Arguments

`object`	Object for which the information should be returned

Value

TRUE, if the object has been imputed and FALSE otherwise.

Author(s)

Michael Scherer

`join.diffMeth,RnBDiffMeth,RnBDiffMeth-method` *join.diffMeth-methods*

Description

Merges two disjoint RnBDiffMeth objects into one. Disjoint here means, that no differential methylation table is specified in both objects.

Usage

```
## S4 method for signature 'RnBDiffMeth,RnBDiffMeth'
join.diffMeth(obj1, obj2)
```

Arguments

`obj1`	`RnBDiffMeth` object. Its base properties will be used to create the joint object this is particularly imported for disk dumped objects as its path will be used and tables from the second object will be copied there
`obj2`	`RnBDiffMeth` object

Value

the merged `RnBDiffMeth` object

Note

Caveat: if disk dumping is enabled the resulting object tables will be stored in the initial location of the first object to be joined I.e. deleting the first object will lead to a broken joined object and deleting the joined object will lead to an broken first object.

Author(s)

Fabian Mueller

Examples

```
library(RnBeads.hg19)
data(small.example.object)
logger.start(fname=NA)
dm1 <- rnb.execute.computeDiffMeth(rnb.set.example,"Sample_Group",c("genes","tiling"))
dm2 <- rnb.execute.computeDiffMeth(rnb.set.example,c("Sample_Group","Treatment"),"promoters")
dm.join1 <- join.diffMeth(dm1,dm2)
#The following joint object is invalid due to missing region type - comparison combinations
is.valid(dm.join1)
dm3 <- rnb.execute.computeDiffMeth(rnb.set.example,"Treatment",c("genes","tiling"))
dm.join2 <- join.diffMeth(dm.join1,dm3)
#After joining the missing information, the new object is valid
is.valid(dm.join2)
```

`limmaP` *limmaP*

Description

applies hierarchical modeling anlalogous to differential expression employed in the `limma` package and returns p-values for differential methylation

Usage

```
limmaP(
  X,
  inds.g1,
  inds.g2 = -inds.g1,
  adjustment.table = NULL,
  fun.conversion = rnb.beta2mval,
  paired = FALSE
)
```

Arguments

`X`	Matrix on which the test is performed for every row
`inds.g1`	column indices of group 1 members
`inds.g2`	column indices of group 2 members
`adjustment.table`	a `data.frame` containing variables to adjust for in the testing
`fun.conversion`	conversion function to transform the beta values into M values. By default, it is the logit function with adjustment for infinity values. See `rnb.beta2mval` for details.
`paired`	should a paired analysis model be used. If so, the first index in `inds.g1` must correspond to the first index in `inds.g2` and so on.

Value

vector of p-values resulting from limma's differential analysis

Note

Requires `limma` package

Author(s)

Fabian Mueller

Examples

```
library(RnBeads.hg19)
data(small.example.object)
logger.start(fname=NA)
meth.mat <- meth(rnb.set.example)
sample.groups <- rnb.sample.groups(rnb.set.example)[[1]]
```

```
p.vals <- limmaP(meth.mat,sample.groups[[1]],sample.groups[[2]])
```

`load.region.subsegment.annotation` *load.region.subsegment.annotation*

Description

For the region annotation of a given `RnBSet` object. Subdivide each region into subsegments by hierarchical clustering on the site distances in a particular region and then splitting the region into subregions consisting of these site clusters. The number of clusters is determined in such way that the mean number of sites per cluster is given by the `ns` parameter.

Usage

```
load.region.subsegment.annotation(rnb.set, annotation.dir)
```

Arguments

`rnb.set`	The `RnBSet object` with subsegments specified in the regions
`annotation.dir`	a directory to load the annotation from. (binary `RData` format.)

Value

invisible `TRUE`

Author(s)

Fabian Mueller

`load.rnb.diffmeth` *load.rnb.diffmeth*

Description

load a saved `RnBDiffMeth` object from disk

Usage

```
load.rnb.diffmeth(path)
```

Arguments

`path`	path of the saved object (a directory containing a corresponding `rnbDiffMeth.RData` file and possibly `rnbDiffMeth_tables` files)

Value

the loaded `RnBDiffMeth` object

Author(s)

Fabian Mueller

`load.rnb.set` *load.rnb.set*

Description

Loading of the `RnBSet` objects with large matrices of type **ff**.

Usage

```
load.rnb.set(path, temp.dir = tempdir())
```

Arguments

`path`	full path of the file or directory. If `archive` is `FALSE`) without an extension.
`temp.dir`	`character` singleton which specifies temporary directory, used while loading

Value

Loaded object

Author(s)

Pavlo Lutsik

`loadLolaDbs` *loadLolaDbs*

Description

Load LOLA databases from disk and merge them

Usage

```
loadLolaDbs(lolaDbPaths)
```

Arguments

`lolaDbPaths`	vector of names of LOLA DB paths to be loaded

Value

LOLA DB list as returned by `LOLA::loadRegionDB`

Author(s)

Fabian Mueller

Examples

```
# download LOLA DB
lolaDest <- tempfile()
dir.create(lolaDest)
lolaDirs <- downloadLolaDbs(lolaDest, dbs="LOLACore")
lolaDb <- loadLolaDbs(lolaDirs[["hg19"]])
```

`logger.argument` *logger.argument*

Description

Reads a command-line argument supplied to a script.

Usage

```
logger.argument(
  arg.names,
  full.name,
  arg.type = "character",
  accepted.values = NULL,
  default = NULL,
  arg.list = commandArgs()
)
```

Arguments

`arg.names`	`character` vector of acceptable argument names. This function scans the provided arguments and performs a case insensitive match.
`full.name`	One-element `character` vector giving the argument's full name or description. This is used in a log message in case of an error.
`arg.type`	Variable type of the argument. Must be one of `"character"`, `"logical"`, `"integer"`, `"double"`, `"numeric"` or `"real"`. The last three types are all synonyms.
`accepted.values`	Vector of accepted values for the argument. This must be of the type given in `arg.type`. Set this to `NULL` if there are no restrictions on the argument values.
`default`	Default value for the argument in case it is not specified. Setting this to `NULL` makes the argument required, that is, an error is generated if the argument is not specified. Set this to `NA` if is not a required argument and it shouldn't default to a specific value. Otherwise, if `accepted.values` is provided, this must be one of its elements.
`arg.list`	Vector of arguments provided at the execution of the script. The arguments should be provided as *name*=*value* pairs.

Details

This is convenience function for reading parameters supplied to the script in the form *name* = *value*. It expects that logging is enabled (see `rnb.options`). The function fails if this condition is not met.

Value

Argument's value, or `NULL` if such is not provided.

Author(s)

Yassen Assenov

Examples

```
n.iterations <- logger.argument("iterations", "number of iterations", "integer",
  accepted.values = 1:100, default = 1L)
logger.close()
```

`logger.getfiles` *logger.getfiles*

Description

Gets the files currently used by the logger.

Usage

```
logger.getfiles()
```

Value

Vector storing the full names of the files that are being used by the logger. This vector contains `NA` as an element if the logger is (also) using the console for its output. If logging functionality is disabled (see `rnb.options`) or the logger is not initialized, this function returns `NULL`.

Author(s)

Yassen Assenov

See Also

`logger.isinitialized` to check if logging is activated; `logger.start` for initializing a logger or starting a section

Examples

```
if (NA %in% logger.getfiles())
  cat("Console logger is enabled\n")
```

`logger.isinitialized` *logger.isinitialized*

Description

Checks if the logger is initialized.

Usage

```
logger.isinitialized()
```

Value

`TRUE` if the logger was initialized and is in use; `FALSE` otherwise.

Author(s)

Yassen Assenov

See Also

`logger.start` for initializing a logger or starting a section

Examples

```
if (!logger.isinitialized())
  logger.start(fname = NA)
```

`logger.machine.name` *logger.machine.name*

Description

Log the machine name the analysis is run on

Usage

```
logger.machine.name()
```

Value

None (invisible `NULL`).

Author(s)

Fabian Mueller

`logger.start` *Log File Management*

Description

Functions for logger management.

Usage

```
logger.start(txt = character(0), fname = NULL)

logger.completed()

logger.close()
```

Arguments

`txt`	Description to add to the log file. The words `STARTED` and `COMPLETED` are prepended to the message upon initialization and completion of the section, respectively.
`fname`	Name of the log file and/or console. Note that at most one file name can be specified. The function `logger.start` normalizes the given name, that is, it converts it to an absolute name. If this parameter is `NA`, logger messages are printed to the console. If it is a two-element vector containing one file name and `NA`, the logger is (re)initialized to print messages both to the given file name and the console. A value of `NULL` (default) indicates the logger should continue using the previously specified file.

Value

None (invisible `NULL`).

Details

`logger.start` initializes the logger and/or starts a new section. `logger.completed` completes the last (innermost) open section in the log. `logger.close` deinitializes the logger. Note that after reinitialization or deinitialization, the information about the current output file, as well as any open sections, is deleted.

Author(s)

Yassen Assenov

See Also

logger.isinitialized

Examples

```
if (!logger.isinitialized())
  logger.start(fname = NA)
logger.start("Tests for Significance")
logger.completed()
logger.close()
```

`logger.status` *Writing text messages to the log file.*

Description

Appends a single-line status message to the log text file. The message is prepended by its type, which is one of `STATUS`, `INFO`, `WARNING` or `ERROR`.

Usage

```
logger.status(txt)

logger.info(txt)

logger.warning(txt)

logger.error(txt, terminate = rnb.getOption("logging.exit.on.error"))
```

Arguments

`txt`	Text to add to the log file. This must be a `character` vector; its elements are concatenated using a single space (`" "`) as a separator.
`terminate`	Flag indicating if the execution is to be terminated after this error message is added to the log.

Value

None (invisible `NULL`).

Author(s)

Yassen Assenov

See Also

`logger.isinitialized`to check if logging is activated; `logger.start` for initializing a logger or starting a section

Examples

```
if (!logger.isinitialized())
  logger.start(fname = NA)
logger.status(c("Reached step", 2))
logger.info(c("Provided email:", rnb.getOption("email")))
```

`logger.validate.file` *logger.validate.file*

Description

Validates the specified file or directory exists. Prints an error or a warning message to the log if it does not exist, it is not of the accepted type or is not accessible.

Usage

```
logger.validate.file(file, is.file = TRUE, terminate = TRUE)
```

Arguments

`file`	Name of file or directory to validate.
`is.file`	Flag indicating if the given name must denote an existing file. If this is `FALSE`, the given name must denote a directory. Set this to `NA` if both types are an acceptable scenario.
`terminate`	Flag indicating if the execution is to be terminated in case the validation fails. This parameter determines if an error message (`terminate` is `TRUE`) or a warning message (`terminate` is `FALSE`) is to be sent to the log when the specified file or directory does not exist, is not of the accepted type or is not accessible.

Value

Whether the validation succeeded or not, invisibly. Note that when `terminate` is `TRUE` and the validation fails, the R session is closed and thus no value is returned.

Author(s)

Yassen Assenov

Examples

```
if (!logger.isinitialized())
  logger.start(fname = NA)
# Validate the current working directory exists
logger.validate.file(getwd(), FALSE)
```

`lolaBarPlot` *lolaBarPlot*

Description

plot a barplot of LOLA enrichment results

Usage

```
lolaBarPlot(
  lolaDb,
  lolaRes,
  scoreCol = "pValueLog",
  orderCol = scoreCol,
  signifCol = "qValue",
  includedCollections = c(),
  pvalCut = 0.01,
  maxTerms = 50,
  colorpanel = sample(rainbow(maxTerms, v = 0.5)),
  groupByCollection = TRUE,
  orderDecreasing = NULL
)
```

Arguments

`lolaDb`	LOLA DB object as returned by `LOLA::loadRegionDB` or `loadLolaDbs`
`lolaRes`	LOLA enrichment result as returned by the `runLOLA` function from the `LOLA` package
`scoreCol`	column name in `lolaRes` to be plotted
`orderCol`	column name in `lolaRes` which is used for sorting the results
`signifCol`	column name of the significance score in `lolaRes`. Should be one of `c("pValueLog", "qValue")`
`includedCollections`	vector of collection names to be included in the plot. If empty (default), all collections are used
`pvalCut`	p-value cutoff to be employed for filtering the results
`maxTerms`	maximum number of items to be included in the plot
`colorpanel`	colors to be used for coloring the bars according to "target" (see `getTargetFromLolaDb`). An empty vector indicates that black will be used for all bars.
`groupByCollection`	facet the plot by collection
`orderDecreasing`	flag indicating whether the value in `orderCol` should be considered as decreasing (as opposed to increasing). `NULL` (default) for automatic determination.

Value

ggplot object containing the plot

Author(s)

Fabian Mueller

Examples

```
library(RnBeads.hg19)
data(small.example.object)
logger.start(fname=NA)
```

```
# compute differential methylation
dm <- rnb.execute.computeDiffMeth(rnb.set.example,pheno.cols=c("Sample_Group","Treatment"))
# download LOLA DB
lolaDest <- tempfile()
dir.create(lolaDest)
lolaDirs <- downloadLolaDbs(lolaDest, dbs="LOLACore")
# perform enrichment analysis
res <- performLolaEnrichment.diffMeth(rnb.set.example,dm,lolaDirs[["hg19"]])
# select the 500 most hypermethylated tiling regions in ESCs compared to iPSCs
# in the example dataset
lolaRes <- res$region[["hESC vs. hiPSC (based on Sample_Group)"]][["tiling"]]
lolaRes <- lolaRes[lolaRes$userSet=="rankCut_500_hyper",]
# plot
lolaBarPlot(res$lolaDb, lolaRes, scoreCol="oddsRatio", orderCol="maxRnk", pvalCut=0.05)
```

`lolaBoxPlotPerTarget` *lolaBoxPlotPerTarget*

Description

plot a boxplot showing LOLA enrichment results per "target" group (see `getTargetFromLolaDb` for an explanation of "target").

Usage

```
lolaBoxPlotPerTarget(
  lolaDb,
  lolaRes,
  scoreCol = "pValueLog",
  orderCol = scoreCol,
  signifCol = "qValue",
  includedCollections = c(),
  pvalCut = 0.01,
  maxTerms = 50,
  colorpanel = c(),
  groupByCollection = TRUE,
  orderDecreasing = NULL,
  scoreDecreasing = NULL
)
```

Arguments

`lolaDb`	LOLA DB object as returned by `LOLA::loadRegionDB` or `loadLolaDbs`
`lolaRes`	LOLA enrichment result as returned by the `runLOLA` function from the `LOLA` package
`scoreCol`	column name in `lolaRes` to be plotted
`orderCol`	column name in `lolaRes` which is used for sorting the results
`signifCol`	column name of the significance score in `lolaRes`. Should be one of `c("pValueLog", "qValue")`

`includedCollections`	vector of collection names to be included in the plot. If empty (default), all collections are used
`pvalCut`	p-value cutoff to be employed for filtering the results
`maxTerms`	maximum number of items to be included in the plot
`colorpanel`	colors to be used for coloring the bars according to "target" (see `getTargetFromLolaDb`). An empty vector indicates that black will be used for all bars.
`groupByCollection`	facet the plot by collection
`orderDecreasing`	flag indicating whether the value in `orderCol` should be considered as decreasing (as opposed to increasing). `NULL` (default) for automatic determination.
`scoreDecreasing`	flag indicating whether the value in `scoreCol` should be considered as decreasing (as opposed to increasing). `NULL` (default) for automatic determination.

Value

ggplot object containing the plot

Author(s)

Fabian Mueller

Examples

```
library(RnBeads.hg19)
data(small.example.object)
logger.start(fname=NA)
# compute differential methylation
dm <- rnb.execute.computeDiffMeth(rnb.set.example,pheno.cols=c("Sample_Group","Treatment"))
# download LOLA DB
lolaDest <- tempfile()
dir.create(lolaDest)
lolaDirs <- downloadLolaDbs(lolaDest, dbs="LOLACore")
# perform enrichment analysis
res <- performLolaEnrichment.diffMeth(rnb.set.example,dm,lolaDirs[["hg19"]])
# select the 500 most hypermethylated tiling regions in ESCs compared to iPSCs
# in the example dataset
lolaRes <- res$region[["hESC vs. hiPSC (based on Sample_Group)"]][["tiling"]]
lolaRes <- lolaRes[lolaRes$userSet=="rankCut_500_hyper",]
# plot
lolaBoxPlotPerTarget(res$lolaDb, lolaRes, scoreCol="oddsRatio", orderCol="maxRnk", pvalCut=0.05)
```

`lolaVolcanoPlot` *lolaVolcanoPlot*

Description

plot a volcano plot showing LOLA enrichment results: LOLA p-value against the log-odds score. Colored by rank

Usage

```
lolaVolcanoPlot(
  lolaDb,
  lolaRes,
  includedCollections = c(),
  signifCol = "qValue",
  colorBy = "maxRnk",
  colorpanel = c()
)
```

Arguments

`lolaDb`	LOLA DB object as returned by `LOLA::loadRegionDB` or `loadLolaDbs`
`lolaRes`	LOLA enrichment result as returned by the `runLOLA` function from the `LOLA` package
`includedCollections`	vector of collection names to be included in the plot. If empty (default), all collections are used
`signifCol`	column name of the significance score in `lolaRes`. Should be one of `c("pValueLog", "qValue")`.
`colorBy`	annotation/column in the the LOLA DB that should be used for point coloring
`colorpanel`	colors to be used for coloring the points

Value

ggplot object containing the plot

Author(s)

Fabian Mueller

Examples

```
library(RnBeads.hg19)
data(small.example.object)
logger.start(fname=NA)
# compute differential methylation
dm <- rnb.execute.computeDiffMeth(rnb.set.example,pheno.cols=c("Sample_Group","Treatment"))
# download LOLA DB
lolaDest <- tempfile()
dir.create(lolaDest)
lolaDirs <- downloadLolaDbs(lolaDest, dbs="LOLACore")
# perform enrichment analysis
res <- performLolaEnrichment.diffMeth(rnb.set.example,dm,lolaDirs[["hg19"]])
# select the 500 most hypermethylated tiling regions in ESCs compared to iPSCs
# in the example dataset
lolaRes <- res$region[["hESC vs. hiPSC (based on Sample_Group)"]][["tiling"]]
lolaRes <- lolaRes[lolaRes$userSet=="rankCut_500_hyper",]
# plot
lolaVolcanoPlot(res$lolaDb, lolaRes, signifCol="qValue")
```

`lump.hg19` *LUMP Support*

Description

The sites used by the LUMP algorithm for estimating immune cell content are stored in an object named `lump.hg19`. This object should not be loaded or otherwise operated on by users. Please refer to the documentation of `rnb.execute.lump` for information on the algorithm and its implementation in **RnBeads**.

Format

`lump.*` is a `list` of non-empty `integer` matrices, one per supported platform. Every `matrix` contains exactly two columns, denoting chromosome index and chromosome-based index, respectively. These indices refer to positions within the probe/site annotation table employed by **RnBeads** for the corresponding platform.

Author(s)

Yassen Assenov

`lump.hg38` *LUMP Support (hg38)*

Description

Those are the same sites as reported in `lump.hg19`, but lifted to ‘hg38‘ with UCSC’s liftOver functionality. This only applies for the CpG-wise sites; i.e. those used for sequencing data sets, since ‘hg38‘ is not supported for array-based data sets.

Format

`lump.*` is a `list` of non-empty `integer` matrices, one per supported platform. Here, only ‘CpG‘ is available for BS datasets.

Author(s)

Michael Scherer

`M,RnBeadRawSet-method` *M-methods*

Description

Extract raw methylated probe intensity from an object of `RnBeadRawSet` class.

Usage

```
## S4 method for signature 'RnBeadRawSet'
M(object, row.names = FALSE)
```

Arguments

`object`	Dataset of interest.
`row.names`	Flag indicating whether the resulting matrix will be assigned row names

Value

`matrix` of the methylated probe intensities

Examples

```
library(RnBeads.hg19)
data(small.example.object)
M.intensity<-M(rnb.set.example)
head(M.intensity)
```

`mask.sites.meth,RnBSet-method` *mask.sites.meth-methods*

Description

Given a logical matrix, sets corresponding entries in the methylation table to NA (masking). Low memory footprint

Usage

```
## S4 method for signature 'RnBSet'
mask.sites.meth(object, mask, verbose = FALSE)
```

Arguments

`object`	Dataset of interest.
`mask`	logical matrix indicating which sites should be masked
`verbose`	if `TRUE` additional diagnostic output is generated

Value

The modified dataset.

`mergeSamples,RnBSet-method`

mergeSamples

Description

Take an RnBSet object and merge methylation and phenotype information given a grouping column in the pheno table coverage is combined by taking the sum of coverages pheno is combined by concatenating entries from all samples

Usage

```
## S4 method for signature 'RnBSet'
mergeSamples(object, grp.col)
```

Arguments

`object`	input RnBSet object
`grp.col`	a column name (string) of `pheno(rnb.set)` that contains unique identifiers for sample groups/replicates to be combined

Details

combines phenotype information, coverage information and methylation information methylation is combined by taking the average. Detection p-values are combined using Fisher's method. For methylation arrays, bead counts are currently not taken into account. objects of class `RnBeadRawSet` are automatically converted to `RnBeadSet`.

Value

the modified RnBSet object

Note

Requires the packages **foreach** and **doParallel**.

Author(s)

Fabian Mueller

Examples

```
library(RnBeads.hg19)
data(small.example.object)
rnb.set.example
rnb.set.merged <- mergeSamples(rnb.set.example,"Cell_Line")
rnb.set.merged
pheno(rnb.set.merged)
```

`meth,RnBSet-method` *meth-methods*

Description

Extracts DNA methylation information (beta values) for a specified set of genomic features.

Usage

```
## S4 method for signature 'RnBSet'
meth(object, type = "sites", row.names = FALSE, i = NULL, j = NULL)
```

Arguments

`object`	dataset of interest.
`type`	`character` singleton. If this is set to `"sites"` (default), DNA methylation information for each available site is returned. Otherwise, this should be one of region types for for which summarized DNA methylation information is computed in the given dataset.
`row.names`	flag indicating if row names are to be generated in the result.
`i`	indices of sites/regions to be retrieved. By default (`NULL`), all will be retrieved.
`j`	indices of samples to be retrieved. By default (`NULL`), all will be retrieved.

Value

`matrix` with methylation beta values.

See Also

`mval` for calculating M values

Examples

```
library(RnBeads.hg19)
data(small.example.object)
## per-site beta-value matrix
mm<-meth(rnb.set.example, row.names=TRUE)
head(mm)
## beta-values for each covered gene
gmm<-meth(rnb.set.example, type="gene", row.names=TRUE)
head(gmm)
```

`mval,RnBSet-method` *mval-methods*

Description

Extracts DNA methylation information (M values) for a specified set of genomic features.

Usage

```
## S4 method for signature 'RnBSet'
mval(object, type = "sites", row.names = FALSE, epsilon = 0)
```

Arguments

`object`	dataset of interest.
`type`	`character` singleton. If this is set to `"sites"` (default), DNA methylation information for each available site is returned. Otherwise, this should be one of region types for for which summarized DNA methylation information is computed in the given dataset.
`row.names`	Flag indicating of row names are to be generated in the result.
`epsilon`	Threshold of beta values to use when adjusting for potential M values close to +infinity or -infinity. See `rnb.beta2mval` for more details.

Value

`matrix` with methylation M values.

See Also

`meth` for extracting methylation beta values

Examples

```
library(RnBeads.hg19)
data(small.example.object)
## per-site M-value matrix
mm<-mval(rnb.set.example, row.names=TRUE)
head(mm)
## M-values for each covered gene
gmm<-mval(rnb.set.example, type="gene", row.names=TRUE)
head(gmm)
```

`nsites,RnBSet-method` *nsites-methods*

Description

Returns the number of sites/regions for a given `RnBSet` object

Usage

```
## S4 method for signature 'RnBSet'
nsites(object, type = "sites")
```

Arguments

`object`	`RnBSet` of interest.
`type`	`character` singleton. If this is set to `"sites"` (default), the number of sites is returned. Otherwise, this should be one of region types for for which the number of regions is returned.

Value

`integer` stating the number of sites/regions. `NA` if the regions have not been summarized yet.

See Also

`meth` Retrieving the matrix of methylation values

Examples

```
library(RnBeads.hg19)
data(small.example.object)
nsites(rnb.set.example)
```

`off,Report-method` *off-methods*

Description

Performs cleanup and/or other finishing activities and closes the specified device, connection, or document.

Usage

```
## S4 method for signature 'Report'
off(.Object)

## S4 method for signature 'ReportPlot'
off(.Object)

## S4 method for signature 'ReportGgPlot'
off(.Object, handle.errors = FALSE)
```

Arguments

`.Object`	Object to be closed.
`handle.errors`	Flag indicating if the method should attempt to catch and process errors (e.g. I/O errors) internally. Setting this to `TRUE` does not guarantee that the method never stops with an error.

Value

The closed object, invisibly.

`parallel.getNumWorkers`

parallel.getNumWorkers

Description

Gets the number of workers used for parallel processing.

Usage

```
parallel.getNumWorkers()
```

Value

Number of workers used for parallel processing; `-1` if parallel processing is not enabled.

Author(s)

Fabian Mueller

Examples

```
parallel.getNumWorkers()
parallel.setup(2)
parallel.getNumWorkers()
parallel.teardown()
parallel.getNumWorkers()
```

`parallel.isEnabled` *parallel.isEnabled*

Description

Checks if whether parallel processing is enabled.

Usage

```
parallel.isEnabled()
```

Value

`TRUE` if multicore processing is enabled, `FALSE` otherwise.

Author(s)

Fabian Mueller

Examples

```
parallel.isEnabled()
parallel.setup(2)
parallel.isEnabled()
parallel.teardown()
parallel.isEnabled()
```

`parallel.setup` *parallel.setup*

Description

Sets up parallel processing. Requires the **foreach** and **doParallel** packages

Usage

```
parallel.setup(...)
```

Arguments

`...`	Parameters for `registerDoParallel` from the **doParallel** package. This allows, for instance, for specificying the number of workers.

Value

`TRUE` (invisible) to indicate that parallelization is set up.

Note

Requires the packages **foreach** and **doParallel**.

Author(s)

Fabian Mueller

Examples

```
parallel.setup(2)
parallel.teardown()
```

`parallel.teardown` *parallel.teardown*

Description

Disables parallel processing.

Usage

```
parallel.teardown()
```

Value

`TRUE`, invisibly.

Author(s)

Fabian Mueller

Examples

```
parallel.getNumWorkers()
parallel.setup(2)
parallel.getNumWorkers()
parallel.teardown()
parallel.getNumWorkers()
```

`performGoEnrichment.diffMeth` *performGoEnrichment.diffMeth*

Description

performs Geno Ontology (GO) enrichment analysis for a given differential methylation table.

Usage

```
performGoEnrichment.diffMeth(
  rnbSet,
  diffmeth,
  ontologies = c("BP", "MF"),
  rank.cuts.region = c(100, 500, 1000),
  add.auto.rank.cut = TRUE,
  rerank = TRUE,
  verbose = TRUE,
  ...
)
```

Arguments

`rnbSet`	RnBSet object for which dirrential methylation was computed
`diffmeth`	RnBDiffMeth object. See `RnBDiffMeth-class` for details.
`ontologies`	GO ontologies to use for enrichment analysis
`rank.cuts.region`	Cutoffs for combined ranking that are used to determine differentially methylated regions
`add.auto.rank.cut`	flag indicating whether an automatically computed cut-off should also be considered.
`rerank`	For deterimining differential methylation: should the ranks be ranked again or should the absolute ranks be used.
`verbose`	Enable for detailed status report
`...`	arguments passed on to the parameters of `GOHyperGParams` from the `GOstats` package

Value

a DiffMeth.go.enrich object (S3) containing the following attributes

`region`	Enrichment information for differential methylation on the region level. See `GOHyperGresult` from the `GOstats` package for furthert details

Author(s)

Fabian Mueller

Examples

```
library(RnBeads.hg19)
data(small.example.object)
logger.start(fname=NA)
dm <- rnb.execute.computeDiffMeth(rnb.set.example,pheno.cols=c("Sample_Group","Treatment"))
res <- performGoEnrichment.diffMeth(rnb.set.example,dm)
```

`performGOenrichment.diffMeth.entrez` *performGOenrichment.diffMeth.entrez*

Description

performs Gene Ontology (GO) enrichment analysis for a list of Entrez identifiers

Usage

```
performGOenrichment.diffMeth.entrez(
  gids,
  uids,
  ontology,
  assembly = "hg19",
  ...
)
```

Arguments

`gids`	gene ids to test (entrez IDs)
`uids`	ids to test against (universe)
`ontology`	which ontology should be used (see `GOHyperGParams` from the `GOstats` package for details)
`assembly`	Genome to be used. One of the following: hg19, mm9, mm10 or rn5
`...`	arguments passed on to the parameters of `GOHyperGParams` from the `GOstats` package

Value

a `GOHyperGresult` object (see the `GOstats` package for further details)

Author(s)

Fabian Mueller

Examples

```
library(RnBeads.hg19)
data(small.example.object)
logger.start(fname=NA)
dm <- rnb.execute.computeDiffMeth(rnb.set.example,pheno.cols=c("Sample_Group","Treatment"))
dmt <- get.table(dm,get.comparisons(dm)[1],"promoters")
annot <- annotation(rnb.set.example,"promoters")
all.promoters <- annot$entrezID
#get the hypermethylated promoters
hyper.promoters <- annot$entrezID[dmt[,"mean.mean.diff"]>0]
result <- performGOenrichment.diffMeth.entrez(hyper.promoters,all.promoters,"BP",assembly="hg19")
```

`performGOEnrichment.diffVar`
performGOEnrichment.diffVar

Description

performs Geno Ontology (GO) enrichment analysis for a given differential variability table.

Usage

```
performGOEnrichment.diffVar(
  rnbSet,
  diffmeth,
  enrich.diffMeth = NULL,
  ontologies = c("BP", "MF"),
  rank.cuts.region = c(100, 500, 1000),
  add.auto.rank.cut = TRUE,
  rerank = TRUE,
  verbose = TRUE,
  ...
)
```

Arguments

`rnbSet`	RnBSet object for which dirrential variability was computed
`diffmeth`	RnBDiffMeth object. See `RnBDiffMeth-class` for details.
`enrich.diffMeth`	Result of `performGOEnrichment.diffMeth`. NULL, if enrichment should only be performed for differential variability.
`ontologies`	GO ontologies to use for enrichment analysis
`rank.cuts.region`	Cutoffs for combined ranking that are used to determine differentially variable regions
`add.auto.rank.cut`	flag indicating whether an automatically computed cut-off should also be considered.
`rerank`	For deterimining differential variability: should the ranks be ranked again or should the absolute ranks be used.
`verbose`	Enable for detailed status report
`...`	arguments passed on to the parameters of `GOHyperGParams` from the `GOstats` package

Value

a DiffMeth.enrich object (S3) containing the following attributes

`region`	Enrichment information for differential variability on the region level. See `GOHyperGresult` from the `GOstats` package for furthert details

Author(s)

Fabian Mueller and Michael Scherer

Examples

```
library(RnBeads.hg19)
data(small.example.object)
logger.start(fname=NA)
dm <- rnb.execute.diffVar(rnb.set.example,pheno.cols=c("Sample_Group","Treatment"))
res <- performEnrichment.diffVar(rnb.set.example,dm)
```

`performLolaEnrichment.diffMeth` *performLolaEnrichment.diffMeth*

Description

performs LOLA enrichment analysis for a given differential methylation table.

Usage

```
performLolaEnrichment.diffMeth(
  rnbSet,
  diffmeth,
  lolaDbPaths,
  rank.cuts.region = c(100, 500, 1000),
  add.auto.rank.cut = TRUE,
  rerank = TRUE,
  verbose = TRUE
)
```

Arguments

`rnbSet`	RnBSet object for which dirrential methylation was computed
`diffmeth`	RnBDiffMeth object. See `RnBDiffMeth-class` for details.
`lolaDbPaths`	LOLA database paths
`rank.cuts.region`	Cutoffs for combined ranking that are used to determine differentially methylated regions
`add.auto.rank.cut`	flag indicating whether an automatically computed cut-off should also be considered.
`rerank`	For deterimining differential methylation: should the ranks be ranked again or should the absolute ranks be used.
`verbose`	Enable for detailed status report

Value

a DiffMeth.lola.enrich object (S3) containing the following attributes

`region`	Enrichment information for differential methylation on the region level. A `data.table` object as returned by the `runLOLA` function from the `LOLA` package for furthert details. Each element will contain different user sets for different rank cutoffs and hyper/hypomethylation events(`userSet` column)
`lolaDb`	The loaded `lolaDb` object containing the merged databases as returned by `loadLolaDbs`

Author(s)

Fabian Mueller

Examples

```
library(RnBeads.hg19)
data(small.example.object)
logger.start(fname=NA)
# compute differential methylation
dm <- rnb.execute.computeDiffMeth(rnb.set.example,pheno.cols=c("Sample_Group","Treatment"))
# download LOLA DB
lolaDest <- tempfile()
dir.create(lolaDest)
lolaDirs <- downloadLolaDbs(lolaDest, dbs="LOLACore")
# perform enrichment analysis
res <- performLolaEnrichment.diffMeth(rnb.set.example,dm,lolaDirs[["hg19"]])
```

`performLolaEnrichment.diffVar` *performLolaEnrichment.diffVar*

Description

performs LOLA enrichment analysis for a given differential variability table.

Usage

```
performLolaEnrichment.diffVar(
  rnbSet,
  diffmeth,
  enrich.diffMeth = NULL,
  lolaDbPaths,
  rank.cuts.region = c(100, 500, 1000),
  add.auto.rank.cut = TRUE,
  rerank = TRUE,
  verbose = TRUE
)
```

Arguments

`rnbSet`	RnBSet object for which differential variability was computed
`diffmeth`	RnBDiffMeth object. See `RnBDiffMeth-class` for details.
`enrich.diffMeth`	Enrichment object as obtained from `performLolaEnrichment.diffMeth`. If it is not provided a new object is created.
`lolaDbPaths`	LOLA database paths
`rank.cuts.region`	Cutoffs for combined ranking that are used to determine differentially variable regions
`add.auto.rank.cut`	flag indicating whether an automatically computed cut-off should also be considered.
`rerank`	For deterimining differential variability: should the ranks be ranked again or should the absolute ranks be used.
`verbose`	Enable for detailed status report

Value

a DiffMeth.lola.enrich object (S3) containing the following attributes

`region`	Enrichment information for differential variability on the region level. A `data.table` object as returned by the `runLOLA` function from the `LOLA` package for further details. Each element will contain different user sets for different rank cutoffs and hyper/hypomethylation events(`userSet` column)
`lolaDb`	The loaded `lolaDb` object containing the merged databases as returned by `loadLolaDbs`

Author(s)

Michael Scherer and Fabian Mueller

Examples

```
library(RnBeads.hg19)
data(small.example.object)
logger.start(fname=NA)
# compute differential methylation
dm <- rnb.execute.diffVar(rnb.set.example,pheno.cols=c("Sample_Group","Treatment"))
# download LOLA DB
lolaDest <- tempfile()
dir.create(lolaDest)
lolaDirs <- downloadLolaDbs(lolaDest, dbs="LOLACore")
# perform enrichment analysis
res <- performLolaEnrichment.diffVar(rnb.set.example,dm,lolaDirs[["hg19"]])
```

`pheno,RnBSet-method` *pheno-methods*

Description

Extracts sample phenotype and/or processing information.

Usage

```
## S4 method for signature 'RnBSet'
pheno(object)
```

Arguments

`object`	Dataset of interest.

Value

Sample annotation information available for the dataset in the form of a `data.frame`.

Examples

```
library(RnBeads.hg19)
data(small.example.object)
pheno(rnb.set.example)
```

`prepareSOFTfileForGEO` *prepareSOFTfileForGEO*

Description

Starting from an `RnBeadSet` object generates a batch submission file for Gene Expression Omnibus series in SOFT format

Usage

```
prepareSOFTfileForGEO(
  rnb.set,
  filename,
  sample.source.col = NULL,
  sample.description.col = NULL,
  sample.title.col = NULL,
  export.cols = seq(ncol(pheno(rnb.set))),
  rnb.set.raw = NULL,
  sample.extra.info = NULL,
  series.info = NULL
)
```

Arguments

`rnb.set`	Object inheriting from class `RnBeadSet` with `"GSE"`.
`filename`	Absolute path or a name of a SOFT file to be generated
`sample.source.col`	`integer` singleton specifying a column in the `pheno` slot of `rnb.set` containing information which will be written into the field `Sample_source_name_ch1` of each sample record
`sample.description.col`	`integer` singleton specifying a column in the `pheno` slot of `rnb.set` containing information which will be written into the field `Sample_desription` of each sample record
`sample.title.col`	`integer` singleton specifying a column in the `pheno` slot of `rnb.set` containing information which will be written into the field `Sample_title` of each sample record. If `NULL`, the result of `samples(rnb.set)` will be used
`export.cols`	`integer` vector specifying columns in the `pheno` slot of `rnb.set` containing information which will be written into the fields `Sample_characteristics_ch1` of each sample record
`rnb.set.raw`	Object inheriting from class `RnBeadSet`
`sample.extra.info`	Optionally, a `list` with elements to be written to all series record. Elements should be character singletons named with valid SOFT labels of a SAMPLE section, e.g.`Sample_extract_protocol`, `Sample_hyb_protocol`, `Sample_label_protocol_ch1`, `Sample_data_processing`, `Sample_contact_name`, `Sample_contact_email` etc.
`series.info`	A `list` with elements to be written to the series record. Elements should be character singletons named `SERIES` (contains a valid GSE identifier for updating an existing series) `Series_title`, `Series_summary`, `Series_type`, `Series_overall_design`, `Series_contributor`, `Series_sample_id`

Details

The code was largely adapted from a similar function in package `lumi` which is due to Pan Du.

Value

`TRUE` on success.

Author(s)

Pavlo Lutsik

`qc,RnBeadSet-method` *qc-methods*

Description

Extracts HumanMethylation quality control information

Usage

```
## S4 method for signature 'RnBeadSet'
qc(object)
```

Arguments

`object`	Dataset of interest.

Value

Quality control information available for the dataset in the form of a `list` with two elements: `Cy3` and `Cy5`.

Examples

```
library(RnBeads.hg19)
data(small.example.object)
qcinf<-dpval(rnb.set.example, row.names=TRUE)
head(qcinf$Cy3)
head(qcinf$Cy5)
```

`read.bed.files` *read.bed.files*

Description

Reads a reduced-representation/whole-genome bisulfite sequencing data set from a set of BED files

Usage

```
read.bed.files(
  base.dir = NULL,
  file.names = NULL,
  sample.sheet = NULL,
  file.names.col = 0,
  assembly = rnb.getOption("assembly"),
  region.types = rnb.region.types.for.analysis(assembly),
  pos.coord.shift = 1L,
  skip.lines = 1,
  sep.samples = rnb.getOption("import.table.separator"),
  merge.bed.files = TRUE,
```

```
    useff = rnb.getOption("disk.dump.big.matrices"),
    usebigff = rnb.getOption("disk.dump.bigff"),
    verbose = TRUE,
    ...
)
```

Arguments

`base.dir`	Directory with BED files contatining processed methylation data
`file.names`	Optional non-empty `character` vector listing the names of the files that should be loaded relative to `base.dir`. If supplied, this vector must not contain `NA` among its elements.
`sample.sheet`	Optional file name containing a table of sample annotation data, or the table itself in the form of a `data.frame` or `matrix`. Only (and all) samples defined in this table will be loaded. The table is expected to contain a column named `"barcode"` that lists the samples' Sentrix barcodes. If such a column is not present, this function searches for columns `"Sentrix_ID"` and `"Sentrix_Position"` (or similar) that build a barcode.
`file.names.col`	Column of the sample sheet which contains the file names (integer singleton). If `NA` an attempt will be made to find a suiting column automatically.
`assembly`	Genome assembly. Defaults to human (`"hg19"`)
`region.types`	`character` vector storing the types of regions for which the methylation information is to be summarized. The function `rnb.region.types` provides the list of all supported regions. Setting this to `NULL` or an empty vector restricts the dataset to site methylation only.
`pos.coord.shift`	The frame shift between the the CpG annotation (1-based) and the coordinates in the loaded BEDs. If BEDs have 0-based coordinates, `pos.coord.shift=1` (default).
`skip.lines`	The number of top lines to skip while reading the BED files
`sep.samples`	`character` singleton used as field separator in the sample sheet file. Default value is taken by the call to `rnb.getOption("import.table.separator")`
`merge.bed.files`	In case multiple BED files are specified for each sample, the flag indicates whether the methylation calls should be merged after reading
`useff`	If `TRUE`, functionality provided by the `ff` package will be used to read the data efficiently.
`usebigff`	flag specifying whether the extended ff functionality should be used (large matrix support for ff)
`verbose`	Flag indicating ifthe messages to the logger should be sent. Note that the logger must be initialized prior to calling this function. Logging is useful for keeping a record of the downloaded and processed samples. Also, informative messages are stored in case of an error.
`...`	Further arguments which are passed to the internal function `read.single.bed` and to `read.table`

Details

To control the BED column assignment, one should also supply arguments to `read.single.bed`.

Value

an object of class `RnBiseqSet`

Author(s)

Pavlo Lutsik

`read.data.dir` *read.data.dir*

Description

Reads in a directory with Illumina Infinium HumanMethylation450 data. The files shoudl be stored as data

Usage

```
read.data.dir(
  dir,
  pheno,
  betas,
  p.values,
  bead.counts,
  sep = rnb.getOption("import.table.separator"),
  verbose = TRUE
)
```

Arguments

`dir`	directory containing the table files
`pheno`	a file containing data sample annotations and phenotypic information
`betas`	a file containing the beta values. If not supplied, the routine will look in dir for a file containing "beta" token in the filename
`p.values`	a file containing the detection p values. If not supplied, the routine will look in dir for a file containing "pval" token in the filename
`bead.counts`	a file containing the bead counts (optional). If not supplied, the routine will look in dir for a file containing "bead" token in the filename
`sep`	character used as field separator in the tables files. Default value is taken by the call to `rnb.getOption("import.table.separator")`
`verbose`	Flag indicating ifthe messages to the logger should be sent. Note that the logger must be initialized prior to calling this function. Logging is useful for keeping a record of the downloaded and processed samples. Also, informative messages are stored in case of an error.

Details

Colnames in all files should match. They will be returned as the samples element of the list.

Value

Object of type `RnBeadSet`.

Author(s)

Pavlo Lutsik

`read.GS.report` *read.GS.report*

Description

Reads in a Genome Studio report, exported as a single file.

Usage

```
read.GS.report(
  gsReportFile,
  pd = NULL,
  sep = rnb.getOption("import.table.separator"),
  keep.methylumi = FALSE,
  verbose = TRUE
)
```

Arguments

`gsReportFile`	location of the GS report file
`pd`	alternative sample annotation, if the `gsReporFile` is missing the sample section as `data.frame` of `character` singleton with the file name
`sep`	character used as field separator in the sample sheet file and in the GS report file (should be identical). Default value is taken by the call to `rnb.getOption("import.table.separat`
`keep.methylumi`	a flag indicating whether the a `MethyLumiSet` object should be returned instead of a `RnBeadRawSet`.
`verbose`	Flag indicating ifthe messages to the logger should be sent. Note that the logger must be initialized prior to calling this function. Logging is useful for keeping a record of the downloaded and processed samples. Also, informative messages are stored in case of an error.

Value

MethylumiSet object with the data from the report

`read.idat.files` *read.idat.files*

Description

Reads a directory of `.idat` files and initializes an object of type `MethyLumiSet`.

Usage

```
read.idat.files(
  base.dir,
  barcodes = NULL,
  sample.sheet = NULL,
  sep.samples = rnb.getOption("import.table.separator"),
  dpval.method = "controls",
  useff = FALSE,
  verbose = TRUE
)
```

Arguments

`base.dir`	Directory that contains the `.idat` files to be read; or a character vector of such directories.
`barcodes`	Optional non-empty `character` vector listing the barcodes of the samples that should be loaded. If supplied, this vector must not contain `NA` among its elements.
`sample.sheet`	Optional file name containing a table of sample annotation data, or the table itself in the form of a `data.frame` or `matrix`. Only (and all) samples defined in this table will be loaded. The table is expected to contain a column named `"barcode"` that lists the samples' Sentrix barcodes. If such a column is not present, this function searches for columns `"Sentrix_ID"` and `"Sentrix_Position"` (or similar) that build a barcode.
`sep.samples`	`character` string used as field separator in the sample sheet file. Default value is taken by the call to `rnb.getOption("import.table.separator")`
`dpval.method`	`character` indicating which method is to be used to compute detection p-values By default, `'controls'` uses the built-in control probes, while the `'pOOBAH'` method from the sesame package is also available.
`useff`	If `TRUE` `ff` package is used to store large matrices on the hard disk
`verbose`	Flag specifying whether the messages to the logger should be sent. Note that the logger must be initialized prior to calling this function. Logging is useful for keeping a record of the downloaded and processed samples. Also, informative messages are stored in case of an error.

Details

If neither `barcodes`, nor `sample.sheet` are specified, the function attempts to locate a file in `base.dir` containing sample annotation information. It fails ifsuch a file cannot be (unambiguously) identified. If both `barcodes` and `sample.sheet` are supplied, only `sample.sheet` is used in loading methylation data. The value of `barcodes` is tested for validity but it is not used as a filter.

Value

Loaded dataset of HumanMethylation450K samples, encapsulated in an object of type `MethyLumiSet`.

Author(s)

Pavlo Lutsik

See Also

`methylumIDAT` in package **methylumi**

`read.idat.files2` *read.idat.files2*

Description

Reads a directory of `.idat` files and initializes an object of type `MethyLumiSet`.

Usage

```
read.idat.files2(
  base.dir,
  barcodes = NULL,
  sample.sheet = NULL,
  sep.samples = rnb.getOption("import.table.separator"),
  load.chunk = NULL,
  keep.methylumi = FALSE,
  verbose = TRUE
)
```

Arguments

`base.dir`	Directory that contains the `.idat` files to be read; or a character vector of such directories.
`barcodes`	Optional non-empty `character` vector listing the barcodes of the samples that should be loaded. If supplied, this vector must not contain `NA` among its elements.
`sample.sheet`	Optional file name containing a table of sample annotation data, or the table itself in the form of a `data.frame` or `matrix`. Only (and all) samples defined in this table will be loaded. The table is expected to contain a column named `"barcode"` that lists the samples' Sentrix barcodes. If such a column is not present, this function searches for columns `"Sentrix_ID"` and `"Sentrix_Position"` (or similar) that build a barcode.
`sep.samples`	character used as field separator in the sample sheet file. Default value is taken by the call to `rnb.getOption("import.table.separator")`
`load.chunk`	`integer` of size one, giving the number of IDAT files which should be loaded in one loading cycle or `NULL`, in which case an attempt will be made to load all files in one go. Should be assigned in case the number of IDATs is more than one thousand.

`keep.methylumi`	a flag indicating whether the a `MethyLumiSet` object should be returned instead of a `RnBeadRawSet`.
`verbose`	Flag indicating ifthe messages to the logger should be sent. Note that the logger must be initialized prior to calling this function. Logging is useful for keeping a record of the downloaded and processed samples. Also, informative messages are stored in case of an error.

Details

If neither `barcodes`, nor `sample.sheet` are specified, the function attempts to locate a file in `base.dir` containing sample annotation information. It fails ifsuch a file cannot be (unambiguously) identified. If both `barcodes` and `sample.sheet` are supplied, only `sample.sheet` is used in loading methylation data. The value of `barcodes` is tested for validity but it is not used as a filter.

Value

Loaded dataset of HumanMethylation450K samples, encapsulated in an object of type `MethyLumiSet`.

Author(s)

Pavlo Lutsik

See Also

`methylumIDAT` in package **methylumi**

`read.sample.annotation` *read.sample.annotation*

Description

Reads Illumina Infinium sample annotation.

Usage

```
read.sample.annotation(fname, sep = rnb.getOption("import.table.separator"))
```

Arguments

`fname`	Name of text file that contains a sample annotation table with a header. This method handles a variety of file formats, including comma-separated values file exported from Genome Studio.
`sep`	One-element `character` used as field separator in the tables file.

Value

Sample annotation table in the form of a `data.frame`, in which every row corresponds to a sample, and every column - to a trait.

Author(s)

Pavlo Lutsik

Examples

```
annotation.file<-system.file("")
sa<-read.sample.annotation(annotation.file)
sa
```

`read.single.bed` *read.single.bed*

Description

reads a BED file with methylation information

Usage

```
read.single.bed(
  file,
  chr.col = 1L,
  start.col = 2L,
  end.col = 3L,
  strand.col = 6L,
  mean.meth.col = 7L,
  coverage.col = 8L,
  c.col = NA,
  t.col = NA,
  is.epp.style = FALSE,
  coord.shift = 0L,
  ffread = FALSE,
  context = "cg",
  ...
)
```

Arguments

`file`	the input BED file
`chr.col`	chromosome column index
`start.col`	start column index
`end.col`	end column index
`strand.col`	strand column index
`mean.meth.col`	mean methylation column index
`coverage.col`	column with coverage information
`c.col`	converted C counts column index
`t.col`	unconverted C counts column index

`is.epp.style`	Flag for custom Broad Epigenome Pipeline (EPP) bed style (columns `"chrom"`, `"start"`, `"end"`, `"methylated_count/total_count"`, `"meth_score_scaled_0_1000"` and `"strand"` in this order). Setting this to `TRUE` overwrites all other parameters except `file`, and also neglects `...`.
`coord.shift`	An integer specifying the coordinate adjustment applied to the start and end coordinates.
`ffread`	Use `ff` package functionality
`context`	prefix for the output rownames
`...`	further arguments to `read.table` or `read.table.ffdf`

Details

Missing columns should be assigned with `NA`. In case `mean.meth.col` is absent at least `coverage.col` and one of `c.col` or `t.col` should be specified.

Value

a `data.frame` or `ff.data.frame` object with DNA methylation and coverage information. The row names are formed by the following convension: `context\.read.delim(file,...)[,chr.col]\.read.delim(file,..`

Author(s)

Pavlo Lutsik

`refFreeEWASP` *refFreeEWASP*

Description

NOTE: This function is deprecated, since the RefFreeEWAS package is not supported and available anymore Applies the reference-free cell-type heterogeneity adjustment model from [1] and returns corrected p-values

Usage

```
refFreeEWASP(
  X,
  inds.g1,
  inds.g2 = -inds.g1,
  adjustment.table = NULL,
  paired = FALSE,
  nboot = 100,
  ignore.na = TRUE,
  rescale.residual = TRUE
)
```

Arguments

`X`	Matrix on which the test is performed for every row
`inds.g1`	column indices of group 1 members
`inds.g2`	column indices of group 2 members
`adjustment.table`	a `data.frame` containing variables to adjust for in the testing
`paired`	should a paired analysis model be used. If so, the first index in `inds.g1` must correspond to the first index in `inds.g2` and so on.
`nboot`	The number of bootstrapping resamples
`ignore.na`	in this case all `NA` containing rows are removed
`rescale.residual`	rescale the residual matrix as z-scores

Value

vector of p-values for the "adjusted" regression coefficients from the Reference-free EWAS model

Note

Requires the package **RefFreeEWAS**.

Author(s)

Pavlo Lutsik

References

1. Houseman, E. Andres, John Molitor, and Carmen J. Marsit. "Reference-Free Cell Mixture Adjustments in Analysis of DNA Methylation Data." Bioinformatics (2014): btu029.

`regionMapping,RnBSet-method`

regionMapping-methods

Description

get the mapping of regions in the RnBSet object to methylation site indices in the RnBSet object

Usage

```
## S4 method for signature 'RnBSet'
regionMapping(object, region.type)
```

Arguments

`object`	Dataset as an object of type inheriting `RnBSet`.
`region.type`	region type. see `rnb.region.types` for possible values

Value

A list containing for each region the indices (as integers) of sites that belong to that region

Author(s)

Fabian Mueller

Examples

```
library(RnBeads.hg19)
data(small.example.object)
logger.start(fname=NA)
promoter.probe.list <- regionMapping(rnb.set.example,"promoters")
#get the number of CpGs per promoter in the dataset:
sapply(promoter.probe.list,length)
```

`regions,RnBSet-method` *regions-methods*

Description

Methylation regions, information for which is present in the `RnBSet` object.

Usage

```
## S4 method for signature 'RnBSet'
regions(object, type = NULL)
```

Arguments

`object`	Dataset of interest.
`type`	Region type(s) of interest as a `character` vector. If this is set to `NULL`, all region types summarized in the object are returned.

Value

Methylation site and region assignment. If `type` is singleton, a `matrix` is returned. The first column corresponds to the methylation context index. The second column is the index of the chromosome in the genome, and the third is the index of the region in the `GRanges` object of the region type annotation. When `length(type)>1`, a list of such matrices is returned for each element of `type`. If `type` is `NULL`, matrices for all summarized region types are returned.

Note

Methylation context index is an integer number denoting the sequence context of the cytosine of interest. Index `1` corresponds to `CpG`, the only supported index in bisulfite sequencing datasets.

Author(s)

Pavlo Lutsik

See Also

`summarized.regions` for all summarized region types in a dataset; `rnb.get.chromosomes` listing all supported chromosomes for a given genome assembly

Examples

```
library(RnBeads.hg19)
data(small.example.object)
head(regions(rnb.set.example))
```

---

reload,RnBDiffMeth-method

reload-methods

Description

reload disk dumped tables. Useful if the table files are manually copied or if the object is loaded again.

Usage

```
## S4 method for signature 'RnBDiffMeth'
reload(
  object,
  save.file,
  disk.path = tempfile(pattern = "diffmeth_", tmpdir = getOption("fftempdir"))
)
```

Arguments

`object`	`RnBDiffMeth` object
`save.file`	location of the ff data saved to disk (i.e. save in save.RData and save.ffData)
`disk.path`	path on the disk for DMTs. can be new or be the same as in the original object

Value

the updated RnBDiffMeth object

Author(s)

Fabian Mueller

Examples

```
library(RnBeads.hg19)
data(small.example.object)
logger.start(fname=NA)
#compute differential methylation
pcols <- c("Sample_Group","Treatment")
tdir <- tempfile(pattern="working")
dm <- rnb.execute.computeDiffMeth(rnb.set.example,pcols,disk.dump=TRUE,disk.dump.dir=tdir)
```

```
#get temporary file names
fn.save.tabs <- tempfile(pattern="saveTables")
fn.save.obj  <- tempfile(pattern="saveObject")
#save the object and the tables to disk
save(dm,file=fn.save.obj)
save.tables(dm,fn.save.tabs)
#delete the object from the workspace
destroy(dm)
rm(dm)
#reload the object and tables
load(fn.save.obj)
dm.new <- reload(dm,fn.save.tabs)
```

`remove.regions,RnBSet-method`
remove.regions-methods

Description

Remove the summarized methylation information for a given region type from an `RnBSet` object.

Usage

```
## S4 method for signature 'RnBSet'
remove.regions(object, region.type)
```

Arguments

`object`	Dataset of interest.
`region.type`	Type of the region annotation for which the summarization should be removed

Value

object of the same class as the supplied one without the summarized methylation information for the specified region type

Examples

```
library(RnBeads.hg19)
data(small.example.object)
summarized.regions(rnb.set.example)
rnb.set.reduced<-remove.regions(rnb.set.example, "genes")
summarized.regions(rnb.set.reduced)
```

`remove.samples,RnBSet-method`

remove.samples-methods

Description

Removes the specified samples from the dataset.

Usage

```
## S4 method for signature 'RnBSet'
remove.samples(object, samplelist)

## S4 method for signature 'RnBeadSet'
remove.samples(object, samplelist)

## S4 method for signature 'RnBeadRawSet'
remove.samples(object, samplelist)
```

Arguments

`object`	Dataset of interest.
`samplelist`	List of samples to be removed in the form of a `logical`, `integer` or `character` vector. If this parameter is `logical`, it is not recycled; its length must be equal to the number of samples in `object`. If it is `integer` or `character`, it must list only samples that exist in the dataset. Specifying sample indices larger than the number of samples, or non-existent sample identifiers results in an error.

Value

The modified dataset.

See Also

`remove.sites` for removing sites or probes from a methylation dataset

Examples

```
library(RnBeads.hg19)
data(small.example.object)
samples(rnb.set.example)
## remove 3 random samples
s2r<-sample.int(length(samples(rnb.set.example)), 3)
rnb.set.f<-remove.samples(rnb.set.example, s2r)
samples(rnb.set.f)
```

`remove.sites,RnBSet-method`

remove.sites-methods

Description

Removes the specified probes from the dataset.

Usage

```
## S4 method for signature 'RnBSet'
remove.sites(object, probelist, verbose = FALSE)

## S4 method for signature 'RnBeadSet'
remove.sites(object, probelist, verbose = TRUE)

## S4 method for signature 'RnBeadRawSet'
remove.sites(object, probelist, verbose = TRUE)
```

Arguments

`object`	Dataset of interest.
`probelist`	List of probes to be removed in the form of a `logical`, `integer` or `character` vector. If this parameter is `logical`, it is not recycled; its length must be equal to the number of probes in `object`. If it is `integer` or `character`, it must list only probes that exist in the dataset. Specifying probe indices larger than the number of probes, or non-existent probe identifiers results in an error.
`verbose`	if `TRUE` additional diagnostic output is generated

Value

The modified dataset.

See Also

`remove.samples` for removing samples from a methylation dataset

Examples

```
library(RnBeads.hg19)
data(small.example.object)
print(rnb.set.example)
## remove 100 random sites
s2r<-sample.int(nrow(sites(rnb.set.example)), 100)
rnb.set.f<-remove.sites(rnb.set.example, s2r)
print(rnb.set.f)
```

`Report-class` *Report Class*

Description

Handler of a generated HTML report. Reports are initialized using the function `createReport`.

Slots

`fname` Name of the file that contains the HTML report.

`dir.conf` Directory that contains configuration files; usually shared between reports.

`dir.data` Directory that contains the generated external lists and tables.

`dir.pngs` Directory that contains the generated figure image files.

`dir.pdfs` Directory that contains the generated figure PDF files.

`dir.high` Directory that contains the generated high-resolution image file.

`sections` Number of sections and subsections currently added to the report.

`opensections` Indices of currently active section and subsections.

`figures` Number of figures currently added to the report.

`tables` Number of selectable tables added to the report.

`references` List of references to be added at the end of the report.

Methods and Functions

`rnb.get.directory` Gets the location of a given report-specific directory.

`rnb.add.section` Generates HTML code for a new section in the report.

`rnb.add.paragraph` Generates HTML code for a new paragraph in the report.

`rnb.add.list` Generates HTML code for a list in the report.

`rnb.add.table` Generates HTML code for a table in the report.

`rnb.add.tables` Generates HTML code for a listing of tables in the report.

`rnb.add.figure` Generates HTML code for a figure in the report.

`rnb.add.reference` Adds a reference item to the report.

`off` Completes the HTML report by adding a reference section (if needed), a footer notice and closing the <body> and `<html>` tags.

Author(s)

Yassen Assenov

`ReportGgPlot-class` *ReportGgPlot Class*

Description

Information about the files created to store one generated plot in a report. Report plots are initialized using the function `createReportGgPlot`. It inherits from the `ReportPlot` class and handling is analogous, except that it contains an additional slot to store a `ggplot` object.

Slots

`ggp` `ggplot` object to be printed

Notes

No device is being opened until `off(reportGgPlot)` is called.

Author(s)

Fabian Mueller

`ReportPlot-class` *ReportPlot Class*

Description

Information about the files created to store one generated plot in a report. Report plots are initialized using the function `createReportPlot`.

Slots

`fname` Relative file name. It does not include path or extension.

`width` Width of the image in inches.

`height` Height of the image in inches.

`create.pdf` Flag indicating if a PDF image is created.

`low.png` Resolution, in dots per inch, used for the figure image.

`high.png` Resolution, in dots per inch, used for the high-resolution image.

`dir.pdf` Directory that contains the generated PDF file.

`dir.png.low` Directory that contains the generated figure image file.

`dir.png.high` Directory that contains the generated high-resolution image file.

Methods and Functions

`get.files` Gets the list of all files that are planned to be generated, or were already generated by the report plot.

`off` Copies the figure to a PNG file (if needed) and closes the device associated with the report plot.

Author(s)

Yassen Assenov

`rnb.add.figure` *rnb.add.figure*

Description

Generates HTML code for a figure in the specified report. A figure is a collection of images (plots), of which only one is visible at any given moment.

Usage

```
rnb.add.figure(
  report,
  description,
  report.plots,
  setting.names = list(),
  selected.image = as.integer(1)
)
```

Arguments

`report`	Report to write the text to.
`description`	Human-readable description of the figure. This must be a non-empty `character` vector. The elements of this vector are concatenated without a separator to form the full description.
`report.plots`	Object of type `ReportPlot`, or a list of such objects.
`setting.names`	List of plot file element descriptors. Every variable elements in the plot file names must be included in this list. Set this to empty list if no variable elements are present, that is, if the figure should present a single report plot.
`selected.image`	Index of plot to be initially selected in the figure.

Value

The modified report.

Author(s)

Yassen Assenov

See Also

`rnb.add.tables` for adding a listing of tables; `Report` for other functions adding contents to an HTML report

---

`rnb.add.list` *rnb.add.list*

Description

Generates HTML code for a list in the specified report.

Usage

```
rnb.add.list(report, txt, type = "u")
```

Arguments

`report`	Report to write the text to.
`txt`	Non-empty list of items to be written. An attribute named `type`, if it exists, specifies the type of the list. See the *Details* section for more information. Every item must be either a nested `list`, denoting a sublist, or a `character` vector (or array), storing the text to be written. Any other objects are coerced to a character type. Elements are concatenated without a separator to form the text for a list item.
`type`	List type to be used for the list and/or its sublists in case the attribute `type` is not specified.

Details

There are two ways to specify a list type: (1) setting a value for the attribute `type` of the list, or (2) using the function's parameter `type`. The value of the function's parameter is used only for lists and sublists that do not contain an attribute named `type`. The following types are supported:

`"o"`	Ordered list using arabic numbers - `1`, `2`, `3`, etc.
`"u"`	Unordered list using bullet points.

Note that every list type must be a one-element `character` vector containing one of the codes listed above. Specifying any other value for list type results in an error.

Value

The modified report, invisibly.

Author(s)

Yassen Assenov

See Also

`Report` for other functions adding contents to an HTML report

Examples

```
report <- createReport("example.html", "Example", init.configuration = TRUE)
recipe <- list("Sift flour in a bowl", "Add sugar and mix", "Add milk and mix")
rnb.add.list(report, recipe, type="o")
```

`rnb.add.paragraph` *rnb.add.paragraph*

Description

Generates HTML code for a new paragraph in the specified report.

Usage

```
rnb.add.paragraph(report, txt, paragraph.class = NULL)
```

Arguments

`report`	Report to write the text to.
`txt`	`character` vector (or array) storing the text to be written. The elements of this vector are concatenated without a separator.
`paragraph.class`	CSS class definition of the paragraph. This must be either `NULL` (default) or one of: `"centered"` This paragraph gives a formula or a short statement. Text is horizontally centered. `"note"` This paragraph describes a note. Text is italic. `"task"` This paragraph describes a task. Text is bold and bright red.

Value

The modified report, invisibly.

Author(s)

Yassen Assenov

See Also

`Report` for other functions adding contents to an HTML report

Examples

```
report <- createReport("example.html", "Example", init.configuration = TRUE)
txt <- "A pessimist is a person who has had to listen to too many optimists."
txt <- c(txt, " <i>Don Marquis</i>")
rnb.add.paragraph(report, txt)
```

`rnb.add.reference` *rnb.add.reference*

Description

Adds a reference item to the given report.

Usage

```
rnb.add.reference(report, txt)
```

Arguments

`report`	Report to add a reference item to.
`txt`	Text of the reference in the form of a non-empty `character` vector. The elements of this vector are concatenated without a separator.

Value

The modified report.

Author(s)

Yassen Assenov

See Also

`rnb.get.reference` for adding citations in the report's text; `Report` for other functions adding contents to an HTML report

Examples

```
report <- createReport("example.html", "Example", init.configuration = TRUE)
txt.reference <- c("Bird A. ", "<i>Nucleic Acids Res.</i> <b>8</b> (1980)")
report <- rnb.add.reference(report, txt.reference)
txt <- c("This was shown in ", rnb.get.reference(report, txt.reference), ".")
rnb.add.paragraph(report, txt)
```

`rnb.add.section` *rnb.add.section*

Description

Generates HTML code for a new section in the specified report.

Usage

```
rnb.add.section(report, title, description, level = 1L, collapsed = FALSE)
```

Arguments

`report`	Report to write the text to.
`title`	Section header. This must be a single-element `character` vector.
`description`	Human-readable paragraph text of the section in the form of a `character` vector. Elements of this vector are concatenated without a separator to form the full description. Set this to `NULL` if the section does not (yet) contain text.
`level`	Section level as a single `integer`. It must be one of `1`, `2` or `3`, denoting section, subsection and sub-subsection, respectively.
`collapsed`	Flag indicating if the contents of this section is to be initially collapsed. Possible values are `TRUE` (the section is not visible), `FALSE` (default, the section is expanded) and `"never"` (the section cannot be collapsed or expanded).

Value

The modified report.

Author(s)

Yassen Assenov

See Also

`Report` for other functions adding contents to an HTML report

Examples

```
report <- createReport("example.html", "Example", init.configuration = TRUE)
report <- rnb.add.section(report, "Introduction", "This is how it's done.")
```

`rnb.add.table` *rnb.add.table*

Description

Generates HTML code for a table in the specified report.

Usage

```
rnb.add.table(
  report,
  tdata,
  row.names = TRUE,
  first.col.header = FALSE,
  indent = 0,
  tag.attrs = c(class = "tabdata"),
  thead = NULL,
  tcaption = NULL,
  na = "<span class=\"disabled\">n/a</span>"
)
```

Arguments

`report`	Report to write the text to.
`tdata`	Matrix or data frame to be presented in HTML form. Column names, if present, are used to define table columns. If this table contains 0 (zero) rows or 0 columns, calling this function has no effect.
`row.names`	Flag indicating if row names should also be printed. If this parameter is `TRUE` and `tdata` defines row names, these are printed in the left-most column and are displayed as header cells. Keep in mind that `data.frames` always define row names.
`first.col.header`	Flag indicating if all cells in the first column must be displayed as header cells. Note that, if both this parameter and `row.names` are `TRUE`, and `tdata` contains row names, the constructed HTML table will have 2 columns of header cells.
`indent`	Default indentation, in number of tabulation characters, to apply to HTML tags. This indentation is also applied to `thead`.
`tag.attrs`	Named `character` vector specifying the list of attributes to be set to the `<table>` element. Setting this to `NULL` or an empty `character` vector disables attributes.
`thead`	`character` vector storing a table header to include. This can, for example, be a `character` that defines column widths. Every element in this vector is written on a separate line, applying the indentation given by `indent`.
`tcaption`	Text to include as a caption below the table, or `NULL` if the table does not contain caption.
`na`	`character` to be used for printing `NA` values in the table. This parameter is not considered when printing `thead` or the table's column names.

Value

The modified report, invisibly.

Author(s)

Yassen Assenov

See Also

`rnb.add.tables` for adding a listing of tables; `Report` for other functions adding contents to an HTML report

---

`rnb.add.tables` *rnb.add.tables*

Description

Generates HTML code for a listing of tables (of which only one is visible at any moment) in the specified report.

Usage

```
rnb.add.tables(
  report,
  tables,
  setting.names,
  selected.table = 1L,
  indent = 2L,
  ...
)
```

Arguments

`report`	Report to write the text to.
`tables`	Non-empty `list` of tables, each one represented by a `data.frame` or `matrix`. The names of this list are used as table identifiers; each one consists of elements separated by underscore character (_).
`setting.names`	List of table name element descriptors. Every variable elements in the table names must be included in this list.
`selected.table`	Index of the table to be initially selected in this listing.
`indent`	Default indentation, in number of tabulation characters, to apply to every table.
`...`	Other parameters passed to `rnb.add.table`.

Value

The modified report.

Author(s)

Yassen Assenov

See Also

`rnb.add.table` for adding a single table to a report; `Report` for other functions adding contents to an HTML report

---

`rnb.annotation.size` *rnb.annotation.size*

Description

Gets the size, in number of genomic elements, of the specified annotation.

Usage

```
rnb.annotation.size(type = "CpG", assembly = "hg19")
```

Arguments

`type`	Name of annotation. Control probe annotations are not accepted.
`assembly`	Genome assembly of interest. See `rnb.get.assemblies` for the list of supported genomes.

Value

`integer` vector showing the number of elements the specified annotation contains per chromosome. The names of the vector are the names of `rnb.get.chromosomes` for the given genome assembly. Chromosomes that are not covered by the annotation have their respective value set to `0` (zero).

Author(s)

Yassen Assenov

See Also

`rnb.region.types` for a list of supported region annotations

Examples

```
library(RnBeads.hg19)
rnb.annotation.size("probes450")
```

`rnb.annotation2data.frame`

rnb.annotation2data.frame

Description

Transform the specified site, probe or region annotation to `data.frame`.

Usage

```
rnb.annotation2data.frame(annotation.table, add.names = TRUE)
```

Arguments

`annotation.table`	Annotation in the form of non-empty `GRangesList` object, as returned by `rnb.get.annotation`.
`add.names`	Flag indicating if element names should be extracted and returned also as a column named `"ID"` in the resulting `data.frame`. Note that element names, if present, are set to be the row names of the table.

Value

Annotation in the form of a single `data.frame`. The columns in this table include, among other, `"Chromosome"`, `"Start"` and `"End"`.

Author(s)

Yassen Assenov

Examples

```
library(RnBeads.hg19)
head(rnb.annotation2data.frame(rnb.get.annotation("probes450")))
```

`rnb.bed.from.segmentation` *rnb.bed.from.segmentation*

Description

This function creates a BED file from the segmentation result of `rnb.execute.segmentation` and stores it on disk.

Usage

```
rnb.bed.from.segmentation(
  rnb.set,
  sample.name,
  type = "final",
  store.path = getwd()
)
```

Arguments

`rnb.set`	An `RnBSet-class` object obtained by executing `rnb.execute.segmentation`.
`sample.name`	The sample name for which segmentation was computed.
`type`	The type of segmentation (`PMDs`, `UMRs`, `LMRs`, `HMDs` or `final`).
`store.path`	Path to which the BED file is to be stored.

Author(s)

Michael Scherer

`rnb.beta2mval` *rnb.beta2mval*

Description

Transforms beta values to M values, adjusting for +infinity and -infinity.

Usage

```
rnb.beta2mval(betas, epsilon = 1e-05)
```

Arguments

`betas`	`numeric` vector or matrix of beta values to be transformed.
`epsilon`	Single `numeric` in the range [0, 0.5], giving the threshold of beta values to use when adjusting for potential M values close to +infinity or -infinity. Setting this parameter to 0 (zero) disables stabilization; in which case M values of -infinity or +infinity could be returned.

Value

The calculated and adjusted M values.

Author(s)

Fabian Mueller

Examples

```
library(RnBeads.hg19)
data(small.example.object)
logger.start(fname=NA)
mvals <- rnb.beta2mval(meth(rnb.set.example))
summary(mvals)
```

`rnb.boxplot.from.segmentation` *rnb.boxplot.from.segmentation*

Description

This function creates a boxplot from the segmentation result of `rnb.execute.segmentation`.

Usage

```
rnb.boxplot.from.segmentation(rnb.set, sample.name, type = "final")
```

Arguments

`rnb.set`	An `RnBSet-class` object obtained by executing `rnb.execute.segmentation`.
`sample.name`	The sample name for which segmentation was computed.
`type`	The type of segmentation (`PMDs`, `UMRs`, `LMRs`, `HMDs` or `final`).

Value

An object of type `ggplot` visualizing the methylation values in the segments.

Author(s)

Michael Scherer

`rnb.build.index` *rnb.build.index*

Description

Creates an HTML index file that contains listing of all available **RnBeads** reports. If no known reports are found in the specified directory, no index is created.

Usage

```
rnb.build.index(
  dir.reports,
  fname = "index.html",
  dir.configuration = "configuration",
  open.index = TRUE
)
```

Arguments

`dir.reports`	Directory that contains HTML reports generated by **RnBeads** modules. If this directory does not exist, is a regular file, is inaccessible, or does not contain any recognizable HTML report files, this function does not generate an HTML index file and produces an error or a warning message.
`fname`	One-element `character` vector specifying the name of the index file to be generated. See the *Details* section for restrictions on the name. The file will be created in `dir.reports`. If such a file already exists, it will be overwritten.
`dir.configuration`	Subdirectory that hosts configuration files shared by the reports. This must be a `character` vector of length one that gives location as a path relative to `dir.reports`. Strong restrictions apply to the path name. See the description of the `createReport` function for more details.
`open.index`	Flag indicating if the index should be displayed after it is created. If this is `TRUE`, `rnb.show.report` is called to open the generated HTML file.

Details

In order to ensure independence of the operating system, there are strong restrictions on the name of the index file. It can consist of the following symbols only: Latin letters, digits, dot (`.`), dash (`-`) and underline (`_`). The extension of the file must be one of `htm`, `html`, `xhtml` or `xml`. The name must not include paths, that is, slash (`/`) or backslash (`\`) cannot be used. In addition, it cannot be any of the recognized **RnBeads** report file names.

Value

Names of all HTML report files that were referenced in the newly generated index, invisibly. The order of the file names is the same as the one they are listed in the index. If no known reports are found in the given directory, the returned value is an empty `character` vector.

Author(s)

Yassen Assenov

See Also

`rnb.run.analysis`, `rnb.initialize.reports`

`rnb.call.destructor` *rnb.call.destructor*

Description

calls the destructor of an RnBSet, RnBeadSet or RnBeadRawSet object conditionally on whether the `enforce.destroy.disk.dumps` option is enabled.

Usage

```
rnb.call.destructor(object, ...)
```

Arguments

`object`	object to be destroyed
`...`	further arguments to the method `destroy`

Value

invisible `TRUE`

Author(s)

Fabian Mueller

`rnb.color.legends` *rnb.color.legends*

Description

Creates a figure in the given report that contains one or more color legends.

Usage

```
rnb.color.legends(
  report,
  legends,
  fprefix = ifelse(is.character(legends), "legend", "legend_"),
  description = "Color legend.",
  setting.names = NULL,
  size.factor = 3
)
```

Arguments

`report`	Report to contain the legend figure. This must be an object of type `Report`.
`legends`	Color legend in the form of a non-empty `character` vector. Element names denote legend labels, and the elements themselves specify colors. This parameter can also be a `list` of color legends. Special restrictions apply to the names of the list elements, see *Details*.
`fprefix`	File name or prefix for the plot files.
`description`	Text of the figure description. See the correponding parameter in `rnb.add.figure` for more details.
`setting.names`	One-element list containing a plot file descriptor, when `legends` is a list. See the corresponding parameter in `rnb.add.figure` for more details. If this is set to `NULL` (default), the list is automatically created using `names(legends)` (when `legends` is a list), or as an empty list (when `legends` is a vector).
`size.factor`	Relative size, in inches of the plots. Legends are displayed in columns of up to 10 items; each column is effectively a square with the specified size.

Details

In case `legends` specifies multiple legends in the form of a list, `names(legends)` are appended to `fprefix` to generate file names. In order to ensure independence of the operating system, there are strong restrictions on these names. They can consist of the following symbols only: Latin letters, digits, dot (`.`), dash (`-`) and underline (`_`).

Value

The modified report.

Author(s)

Yassen Assenov

`rnb.combine.arrays` *Combine array-based datasets*

Description

Concatenates two array-based datasets focusing on the common probes.

Usage

```
rnb.combine.arrays(dataset1, dataset2, type = "common")
```

Arguments

`dataset1`	First input dataset as an object of type inheriting `RnBeadSet`.
`dataset2`	Second input dataset as an object of type inheriting `RnBeadSet`.
`type`	Type of the combine operation as a character singleton, one of "common", "all.x", "all.y" and "all".

Details

Sample annotation tables This method expects that the sample annotation tables of the two datasets have identical structures.

Genome assembly This method expects that the two datasets target the same genome assembly.

Platform The platform of the combined dataset is the most recent among the platforms of the input datasets.

Intensity values The combined dataset is of type `RnBeadRawSet` only when both input datasets are of this type. Otherwise, any intensity value data is ignored.

Probes Only the common probes are included in the resulting dataset.

Regions Regions summarized in any of the input datasets are ignored. In the resulting dataset, regions are summarized as specified in the analysis option `"region.types"`.

Quality control data QC data in the input datasets is ignored. The combined dataset includes no data on QC probe intensities.

Infered covariates Inferred covariates in the input datasets are ignored. The combined dataset includes no data on inferred covariates.

Disk dumping The combined dataset stores big tables on disk when the analysis option `"disk.dump.big.matrices"` is enabled.

Value

Combined dataset as an object of type inheriting `RnBeadSet`.

Author(s)

Yassen Assenov

`rnb.combine.seq` *rnb.combine.seq*

Description

Initial implementation of the combine method for sequencing datasets.

Usage

```
rnb.combine.seq(x, y, type = "common")
```

Arguments

`x`	An object of type `RnBiseqSet-class` used for concatenation
`y`	Another object of type `RnBiseqSet-class` used for concatenation
`type`	A character representing the type of combination. Needs to be one of `"common"`, `all.x`, `all.y` or `all`.

Details

The type parameters determines the mode of combination:

- `"common"` The intersection between the sites present in the two datasets is used for the new dataset.
- `"all.x"` All sites present in x are used.
- `"all.y"` All sites present in y are used.
- `"all"` The union between the sites of both datasets is used.

Value

An `RnBiseqSet-class` object with combined information

`rnb.execute.age.prediction` *rnb.execute.age.prediction*

Description

Performs age prediction by either the specified predictor in the option `inference.age.prediction.predictor` or by the corresponding predefined predictor.

Usage

```
rnb.execute.age.prediction(object)
```

Arguments

`object`	a `RnBSet` object for which age prediction should be performed

Value

modified `RnBSet` object

Author(s)

Michael Scherer

`rnb.execute.batch.qc` *rnb.execute.batch.qc*

Description

Computation of correlations and permutation-based p-values for detecting quality-associated batch effects.

Usage

```
rnb.execute.batch.qc(rnb.set, pcoordinates, permutations = NULL)
```

Arguments

`rnb.set`	HumanMethylation450K dataset as an object of type `RnBeadSet`.
`pcoordinates`	Coordinates of the samples of `rnb.set` in the principal components space, as returned by `rnb.execute.dreduction`.
`permutations`	Matrix of sample index permutations, as returned by `rnb.execute.batcheffects`. If this parameter is `NULL`, permutation-based p-values are not calculated.

Value

`NULL` if no principal components for batch analysis are specified (`rnb.getOption("exploratory.principal.compone` `== 0`); otherwise, a hierarchical structure of matrices in the form of a nested list. The root branches are represented by the elements `"correlations"` and `"pvalues"`. Every element is a list of control probe types; each type is in turn a list of up to two matrices of correlations between probe values and principal components - one for the probes on the green channel and one for the red channel. Note that the `"pvalues"` branch is not returned when `permutations` is `NULL`.

Author(s)

Pavlo Lutsik

`rnb.execute.batcheffects` *rnb.execute.batcheffects*

Description

Performs tests for association between traits and principal components.

Usage

```
rnb.execute.batcheffects(rnb.set, pcoordinates = NULL)
```

Arguments

`rnb.set`	Methylation dataset as an object of type inheriting `RnBSet`.
`pcoordinates`	Coordinates of the samples of `rnb.set` in the principal components space, as returned by `rnb.execute.dreduction`.

Value

Results of attempted tests for associations in the form of a list with up to three elements:

`"permutations"` `integer` matrix of index permutations. The number of rows in the matrix is N - the number of samples in `rnb.set`. Every column in this matrix denotes a sample permutation; the first column is the sequence 1 to N. This element is included only when `rnb.getOption("exploratory.correlation.permutations")` is non-zero and there are numeric traits to be tested.

`"pc"` List of four matrices named `"failures"`, `"tests"`, `"correlations"` and `"pvalues"`. The rows in each of these matrices correspond to the first several principal components, and the columns - to selected traits. This element is not included in the returned list when `pcoordinates` is `NULL`.

`"traits"` List of four square symmetric matrices named `"failures"`, `"tests"`, `"correlations"` and `"pvalues"`, containing information about the performed tests for pairwise trait association. This element is included only if two or more traits were tested.

Author(s)

Yassen Assenov

See Also

`rnb.run.exploratory` for running the whole exploratory analysis module

Examples

```
library(RnBeads.hg19)
data(small.example.object)
regs <- c("sites", summarized.regions(rnb.set.example))
dreduction <- function(x) rnb.execute.dreduction(rnb.set.example, x)
pcoordinates <- lapply(regs, dreduction)
names(pcoordinates) <- regs
result <- rnb.execute.batcheffects(rnb.set.example, pcoordinates)
```

`rnb.execute.clustering`

rnb.execute.clustering

Description

Performs hierarchical clustering on the samples of the given dataset using multiple distance metrics and agglomeration methods for a single given region type.

Usage

```
rnb.execute.clustering(rnb.set, region.type = "sites")
```

Arguments

`rnb.set`	Methylation dataset as an object of type inheriting `RnBSet`.
`region.type`	the clustering is performed on methylation levels from regions of that type. see `rnb.region.types` for possible values.

Value

List of clustering results, whereby each element is an object of type `RnBeadClustering`. In case clustering cannot be performed, the return value is `NULL`. Reasons for a failure include, among others, the case when `rnb.set` contains less than 3 samples, or undefined distances between a pair of samples due to (too many) missing values in the respective methylation matrix.

Author(s)

Yassen Assenov

Examples

```
library(RnBeads.hg19)
data(small.example.object)
results <- rnb.execute.clustering(rnb.set.example, "promoters")
# List applied dissimilarity metrics
sapply(results, slot, "dissimilarity")
# List applied clustering algorithms
str(lapply(results, slot, "algorithm"))
```

`rnb.execute.clustering.all`
rnb.execute.clustering.all

Description

Performs hierarchical clustering on the samples of the given dataset using multiple distance metrics and agglomeration methods for all suggested site and region types.

Usage

```
rnb.execute.clustering.all(rnb.set)
```

Arguments

`rnb.set`	Methylation dataset as an object of type inheriting `RnBSet`.

Value

List of list of clustering results; each element corresponds to one region type and is a list of objects of type `RnBeadClustering`.

Author(s)

Fabian Mueller

See Also

`rnb.execute.clustering` for performing clustering using a single site or region type.

`rnb.execute.computeDiffMeth`

rnb.execute.computeDiffMeth

Description

computes differential methylation

Usage

```
rnb.execute.computeDiffMeth(
  x,
  pheno.cols,
  region.types = rnb.region.types.for.analysis(x),
  covg.thres = rnb.getOption("filtering.coverage.threshold"),
 pheno.cols.all.pairwise = rnb.getOption("differential.comparison.columns.all.pairwise"),
  columns.pairs = rnb.getOption("columns.pairing"),
  columns.adj = rnb.getOption("covariate.adjustment.columns"),
  adjust.sva = rnb.getOption("differential.adjustment.sva"),
  pheno.cols.adjust.sva = rnb.getOption("inference.targets.sva"),
  adjust.celltype = rnb.getOption("differential.adjustment.celltype"),
  skip.sites = !rnb.getOption("analyze.sites"),
  disk.dump = rnb.getOption("disk.dump.big.matrices"),
  disk.dump.dir = tempfile(pattern = "diffMethTables_"),
  ...
)
```

Arguments

`x`	RnBSet object
`pheno.cols`	column names of the pheno slot in `x` on which the dataset should be partitioned. Those columns are required to be factors or logical. In case of factors, each group in turn will be compared to all other groups
`region.types`	which region types should be processed for differential methylation
`covg.thres`	coverage threshold for computing the summary statistics. See `computeDiffTab.extended.site` for details.
`pheno.cols.all.pairwise`	integer or character vector specifying the colomns of `pheno(x)` on which all pairwise comparisons should be conducted. A value of `NULL` (default) indicates no columns.
`columns.pairs`	argument passed on to `rnb.sample.groups`. See its documentation for details.
`columns.adj`	Column names or indices in the table of phenotypic information to be used for confounder adjustment in the differential methylation analysis.
`adjust.sva`	flag indicating whether the adjustment table should also contain surrogate variables (SVs) for the given target variable.
`pheno.cols.adjust.sva`	Column names or indices in the table of phenotypic information to be used for SVA adjustment in the differential methylation analysis.

`adjust.celltype`	flag indicating whether the resulting table should also contain estimated celltype contributions. See `rnb.execute.ct.estimation` for details.
`skip.sites`	flag indicating whether differential methylation in regions should be computed directly and not from sites. This leads to skipping of site-specific differential methylation
`disk.dump`	Flag indicating whether the resulting differential methylation object should be file backed, ie.e the matrices dumped to disk
`disk.dump.dir`	disk location for file backing of the resulting differential methylation object. Only meaningful if `disk.dump=TRUE`. must be a character specifying an NON-EXISTING valid directory.
`...`	arguments passed on to binary differential methylation calling. See `computeDiffTab.extended.sit` for details.

Value

an `RnBDiffMeth` object. See class description for details.

Author(s)

Fabian Mueller

Examples

```
library(RnBeads.hg19)
data(small.example.object)
logger.start(fname=NA)
dm <- rnb.execute.computeDiffMeth(rnb.set.example,pheno.cols=c("Sample_Group","Treatment"))
get.comparisons(dm)
```

`rnb.execute.context.removal`

rnb.execute.context.removal

Description

Removes all probes that belong to specific context from the given dataset.

Usage

```
rnb.execute.context.removal(
  rnb.set,
  contexts = rnb.getOption("filtering.context.removal")
)
```

Arguments

`rnb.set`	Methylation dataset as an object of type `RnBeadSet`.
`contexts`	Probe contexts to be filtered out.

Value

List of three or four elements:

`"dataset.before"` Copy of `rnb.set`.

`"dataset"` The (possibly modified) `RnBeadSet` object after performing the missing value removal.

`"filtered"` `integer` vector storing the indices of all removed probes in `dataset.before`.

`"contexts"` The value of the parameter `contexts`.

Author(s)

Yassen Assenov

Examples

```
library(RnBeads.hg19)
data(small.example.object)
contexts.to.ignore <- c("CC", "CAG", "CAH")
rnb.set.filtered <- rnb.execute.context.removal(rnb.set.example, contexts.to.ignore)$dataset
identical(rnb.set.example, rnb.set.filtered) # FALSE
```

`rnb.execute.cross.reactive.removal` *rnb.execute.cross.reactive.removal*

Description

Removes all probes defined as cross-reactive from the given dataset.

Usage

```
rnb.execute.cross.reactive.removal(rnb.set)
```

Arguments

`rnb.set`	Methylation dataset as an object of type inheriting `RnBeadSet`.

Value

`list` of four elements:

`"dataset.before"` Copy of `rnb.set`.

`"dataset"` The (possibly) modified dataset object after removing probes that have a high likelihood of cross-hybridization.

`"filtered"` `integer` vector storing the indices (in beta matrix of the unfiltered dataset) of all removed probes.

Author(s)

Yassen Assenov

Examples

```
library(RnBeads.hg19)
data(small.example.object)
rnb.set.filtered <- rnb.execute.cross.reactive.removal(rnb.set.example)$dataset
identical(meth(rnb.set.example), meth(rnb.set.filtered)) # FALSE
```

`rnb.execute.ct.estimation`

rnb.execute.ct.estimation

Description

Perform the estimation of the cell type contributions in each analyzed sample.

Usage

```
rnb.execute.ct.estimation(
  rnb.set,
  cell.type.column = NA,
  test.max.markers = NA,
  top.markers = 500,
  method = "houseman1",
  verbose = TRUE
)
```

Arguments

`rnb.set`	object of class `RnBSet`
`cell.type.column`	integer index or character identifier of a column in sample annotation table of `rnb.set` which gives the mapping of samples to reference cell types
`test.max.markers`	Maximal amount of CpG positions to use for marker selection. If this option is set to `NA` or `NULL`, all sites are considered. Please take into account the extended computation time in such a case.
`top.markers`	the number of markers to select
`method`	algorithm used for estmation of the cell type contributions
`verbose`	flag specifying whether diagnostic output should be written to the console or to the RnBeads logger in case the latter is initialized

Details

The only supported method is the one from Houseman et al BMC Bioinformatics 2012

Value

object of class `CellTypeInferenceResult`

Author(s)

Pavlo Lutsik

`rnb.execute.diffVar` *rnb.execute.diffVar*

Description

This routine computes sites that are differentially variable between two sample groups specified as the column name in the phenotypic table.

Usage

```
rnb.execute.diffVar(
  rnb.set,
  pheno.cols = rnb.getOption("differential.comparison.columns"),
  region.types = rnb.region.types.for.analysis(rnb.set),
  columns.adj = rnb.getOption("covariate.adjustment.columns"),
  adjust.celltype = rnb.getOption("differential.adjustment.celltype"),
  disk.dump = rnb.getOption("disk.dump.big.matrices"),
  disk.dump.dir = tempfile(pattern = "diffMethTables_")
)
```

Arguments

`rnb.set`	Object of type `RnBSet` on which differential variability analysis should be conducted
`pheno.cols`	Column names used to define the classes, whose methylation variability should be compared with each other
`region.types`	Regions types to be used for the analysis. Defaults to the results given by rnb.region.types.for.analysis of the given RnBSet.
`columns.adj`	Column names or indices in the table of phenotypic information to be used for confounder adjustment in the differential variability analysis.
`adjust.celltype`	Flag indicating whether the resulting table should also contain estimated celltype contributions. See `rnb.execute.ct.estimation` for details.
`disk.dump`	Flag indicating whether the resulting differential methylation object should be file backed, ie.e the matrices dumped to disk
`disk.dump.dir`	disk location for file backing of the resulting differential methylation object. Only meaningful if `disk.dump=TRUE`.

Value

Object of type `RnBDiffMeth` containing information about the differential variability analysis.

Author(s)

Michael Scherer

`rnb.execute.dreduction`

rnb.execute.dreduction

Description

Performs principal component analysis (PCA) and multi-dimensional scaling (MDS) of the samples in the given methylation dataset.

Usage

```
rnb.execute.dreduction(rnb.set, target = "sites")
```

Arguments

`rnb.set`	Methylation dataset as an object of type inheriting `RnBSet`. This dataset must contain at least four samples.
`target`	`character` singleton specifying the level of DNA methylation infromation. If this is `"sites"`, the DNA methylation information for the individual sites or probes is analyzed. Otherwise, this should be one of the supported region types, as returned by `rnb.region.types`.

Details

Row names in the returned matrices are sample identifiers, determined based on the package option `"identifiers.column"`. See *RnBeads Options* for more information on this option.

Value

Results of the dimension reduction in the form of a list with the following elements:

`pca` Results of the PCA as returned by the function `prcomp`.

`mds` List of two elements - `"manhattan"` and `"euclidean"`, each of which is a two-column `matrix` storing the coordinates of the samples in a two-dimensional space. The matrices are computed using the function `isoMDS`.

Author(s)

Yassen Assenov

See Also

`rnb.run.exploratory` for running the whole exploratory analysis module

Examples

```
library(RnBeads.hg19)
data(small.example.object)
regs <- c("sites", summarized.regions(rnb.set.example))
dreduction <- function(x) rnb.execute.dreduction(rnb.set.example, x)
pcoordinates <- lapply(regs, dreduction)
names(pcoordinates) <- regs
str(pcoordinates)
```

`rnb.execute.export.csv` *rnb.execute.export.csv*

Description

Exports (selected) methylation tables of the given dataset to comma-separated value files.

Usage

```
rnb.execute.export.csv(
  rnb.set,
  output.location,
  region.types = rnb.getOption("export.types")
)
```

Arguments

`rnb.set`	Methylation dataset as an object of type inheriting `RnBSet`.
`output.location`	`character` or `Report` specifying the output directory. If this is a report, the output directory is set to be a subdirectory named `csv` of the report's data directory. Set this parameter to the empty string (`""`) or `NA` to use the current working directory. If the given path does not exist, this function attempts to create it.
`region.types`	`character` vector indicating region types to be exported.

Details

The names of the generated output files are formed by the prefix `"betas_"`, followed by a number between 1 and `length(region.types)`. The extension is `.csv` or `.csv.gz`, depending on the value of the **RnBeads** option `"gz.large.files"`. Any such files that already exist in the output directory, are overwritten.

There are several reasons why a certain output file cannot be (fully) generated. Examples for failures are listed below:

- The corresponding region type is invalid.
- The corresponding region type is not supported by the dataset. If the type is loaded in **RnBeads**, use the `summarize.regions` method prior to calling this function, in order to include the support of this region type in the dataset.
- Due to security restrictions, the creation of files in the output directory is not allowed.
- A file or directory with the same name exists and cannot be overwritten.
- The disk is full or the user quota is exceeded.

Value

`character` vector containing the names of the files to which data were exported; prepended by `output.location`. In case a certain region type could not be exported (see the *Details* section), the corresponding element of this vector is `NA`.

Author(s)

Yassen Assenov

Examples

```
library(RnBeads.hg19)
data(small.example.object)
rnb.execute.export.csv(rnb.set.example, "", summarized.regions(rnb.set.example))
```

`rnb.execute.filter.summary`

rnb.execute.filter.summary

Description

Calculates a table summarizing the effect of the applied filtering procedures.

Usage

```
rnb.execute.filter.summary(old.set, new.set)
```

Arguments

`old.set`	Methylation dataset before filtering as an object of type inheriting `RnBSet`.
`new.set`	Methylation dataset after filtering as an object of type inheriting `RnBSet`.

Details

This function expects that the sites and samples in `new.set` are subsets of the sites and samples in `old.set`, respectively. If this is not the case, it exists with an error.

Value

`matrix` summarizing the number of removed and retained sites, samples, and (optionally) reliable and unreliable measurements.

Author(s)

Yassen Assenov

See Also

`rnb.run.preprocessing` for running the whole preprocessing module

`rnb.execute.gender.prediction`
rnb.execute.gender.prediction

Description

Deprecated function name, now called `rnb.execute.sex.prediction`.

Usage

```
rnb.execute.gender.prediction(rnb.set)
```

Arguments

`rnb.set`	Methylation dataset after running the sex prediction step, as an object of type `RnBSet`.

Value

The possibly modified dataset. If sex could be predicted, the sample annotation table is enriched with

See Also

rnb.execute.sex.prediction

`rnb.execute.genomewide`
Genome-wide methylation level

Description

Computes genome-wide methylation levels per sample.

Usage

```
rnb.execute.genomewide(dataset)
```

Arguments

`dataset`	Methylation dataset to study, provided as an object of type inheriting `RnBSet`.

Value

`vector` of values in the range `[0, 1]`, storing the average beta values per sample.

Author(s)

Yassen Assenov

`rnb.execute.greedycut` *rnb.execute.greedycut*

Description

Executes the Greedycut procedure for probe and sample filtering based on the detection p-values, and calculates statistics on its iterations.

Usage

```
rnb.execute.greedycut(
  rnb.set,
  pval.threshold = rnb.getOption("filtering.greedycut.pvalue.threshold"),
  min.coverage = rnb.getOption("filtering.coverage.threshold"),
  rc.ties = rnb.getOption("filtering.greedycut.rc.ties")
)
```

Arguments

`rnb.set`	HumanMethylation450K dataset as an object of type `RnBeadSet`.
`pval.threshold`	The P-value threshold. For further information, see the option `"filtering.greedycut.pvalue.thr` in `rnb.options`.
`min.coverage`	The coverage threshold. For further information, see the option `"filtering.coverage.threshold"` in `rnb.options`.
`rc.ties`	Flag indicating what the behaviour of the algorithm should be in case of ties between values of rows (probes) and columns (samples). See the corresponding parameter in `greedycut.filter.matrix` for more details.

Value

`NULL` if `rnb.set` does not contain a matrix of detection p-values, or if all p-values denote reliable measurements. Otherwise, a list of the following elements:

"infos" Table summarizing the iterations of the algorithm, as returned by `greedycut.filter.matrix`.

"statistics" Additional statistics on all iterations, as returned by `greedycut.get.statistics`.

"iteration" Number of Greedycut iterations + 1 applied to the dataset, that is, a value of 1 indicates that the dataset was not modified.

"sites" Indices of all sites to be removed.

"samples" Indices of all samples to be removed.

Author(s)

Yassen Assenov

Examples

```
library(RnBeads.hg19)
data(small.example.object)
greedy.result <- rnb.execute.greedycut(rnb.set.example)
# Number of applied iterations
greedy.result$iteration
```

`rnb.execute.high.coverage.removal` *rnb.execute.high.coverage.removal*

Description

Removes methylation sites with a coverage larger than 100 times the 95-percentile of coverage in each sample.

Usage

```
rnb.execute.high.coverage.removal(rnb.set)
```

Arguments

`rnb.set`	Methylation dataset as an object of type inheriting `RnBiseqSet`.

Value

`list` of two elements:

`"dataset"` The (possibly) modified dataset after retaining sites on autosomes only.

`"filtered"` `integer` vector storing the indices of all removed sites.

Author(s)

Fabian Mueller

`rnb.execute.high.dpval.masking` *rnb.execute.high.dpval.masking*

Description

Replaces all low coverage sites by `NA`.

Usage

```
rnb.execute.high.dpval.masking(rnb.set, dpval.threshold = 0.05)
```

Arguments

`rnb.set`	Methylation dataset as an object of type inheriting `RnBeadSet`.
`dpval.threshold`	Threshold for maximal acceptable detection p-value, given as a non-negative `numeric` value between 0 and 1. All methylation measurements with detection p-value than this threshold are set to `NA`. If this parameter is `0`, calling this method has no effect.

Value

List of three elements:

`"dataset.before"` Copy of `rnb.set`.

`"dataset"` The (possibly) modified dataset after retaining sites on autosomes only.

`"mask"` A logical matrix of dimension `meth(rnb.set,type="sites")` indicating which methylation values have been masked

Author(s)

Fabian Mueller

`rnb.execute.import` *rnb.execute.import*

Description

Loads the data from the specified type and encapsulates it in either an `RnBSet`-inheriting object

Usage

```
rnb.execute.import(
  data.source,
  data.type = rnb.getOption("import.default.data.type"),
  dry.run = FALSE,
  verbose = TRUE
)
```

Arguments

`data.source`	non-empty `character` vector or `list` specifying the location of the data items. The expected format depends on the `data.type` that is given. See the *Details* section.
`data.type`	type of the input data; must be one of `"idat.dir"`, `"data.dir"`, `"data.files"`, `"GS.report"`, `"GEO"` or `"rnb.set"`.
`dry.run`	if `TRUE` and `data.type` is `"bs.bed.dir"`, only a test data import is performed and first 10,000 lines are read from each BED file
`verbose`	flag specifying whether diagnostic output should be written to the console or to the RnBeads logger in case the latter is initialized

Details

The interpretation of `data.source` depends on the value of `data.type` and is summarized in the following table:

`data.type`	**Type of** `data.source`	**Maximal length of** `data.source`	**Interpretation**
`"infinium.idat.dir"`	`list` or `character`	2	(1) Directory containing IDAT
`"infinium.data.dir"`	`character`	1	Directory containing data table
`"infinium.data.files"`	`character`	2..4	The character vector should co

"infinium.GS.report"	character	1	Genome Studio report file
"infinium.GEO"	character	1	GEO identifier or downloaded
"bs.bed.dir"	list or character	1..3	(1) Directory with BED files ea
"rnb.set"	RnBSet	1	object of class inheriting from

Value

Loaded data as an object of type `RnBSet` (when the input data type is `"data.dir"`, `"data.files"` or `"GEO"`) or of type `MethyLumiSet` (when the data type is `"idat.dir"` or `"GS.report"`).

Author(s)

Pavlo Lutsik

See Also

`read.data.dir`, `read.idat.files`, `read.GS.report`, `rnb.read.geo`, `read.bed.files` #'

Examples

```
# Directory where your data is located
data.dir <- "~/RnBeads/data/Ziller2011_PLoSGen_450K"
idat.dir <- file.path(data.dir, "idat")
sample.annotation <- file.path(data.dir, "sample_annotation.csv")
data.source <- c(idat.dir, sample.annotation)
rnb.set <- rnb.execute.import(data.source = data.source, data.type = "idat.dir")
```

`rnb.execute.imputation` *rnb.execute.imputation*

Description

Removes missing methylation values in the methylation matrix of the given object

Usage

```
rnb.execute.imputation(
  rnb.set,
  method = rnb.getOption("imputation.method"),
  update.ff = TRUE,
  ...
)
```

Arguments

`rnb.set`	Dataset object inheriting from `RnBSet`.
`method`	Imputation method to be used, must be one of `"mean.cpgs"`, `"mean.samples"`, `"random"`, `"knn"`, `"median.cpgs"`, `"median.samples"`, or `"none"`.
`update.ff`	flag indicating if the disk based matrices should be updated. Should be set to FALSE, if methylation matrix should only temporarly be changed. If this value is FALSE, the region level methylation values are not updated and only the site-wise matrix is changed temporarly.
`...`	Optional arguments passed to knn.imputation

Details

Imputes missing values by applying on the following methods:

mean.cpgs: missing values are inferred as the average methylation value from all other (non-mising) CpGs in this sample

mean.samples: missing values are inferred as the average methylation value from all other (non-mising) values at this CpG sites in all other samples

random: missing values are inferred by randomly selecting a (non-missing) methylation value from any other sample at this CpG site

knn: missing values are inferred by k-nearest neighbors imputation (see **impute**)

median.cpgs: missing values are inferred as the median methylation value from all other (non-mising) CpGs in this sample

median.samples: missing values are inferred as the median methylation value from all other (non-mising) values at this CpG sites in all other samples

none: imputation should not be performed

Value

The modified rnb.set object without missing methylation values.

Author(s)

Michael Scherer

`rnb.execute.low.coverage.masking` *rnb.execute.low.coverage.masking*

Description

Replaces all low coverage sites by `NA`.

Usage

```
rnb.execute.low.coverage.masking(
  rnb.set,
  covg.threshold = rnb.getOption("filtering.coverage.threshold")
)
```

Arguments

`rnb.set`	Methylation dataset as an object of type inheriting `RnBSet`.
`covg.threshold`	Threshold for minimal acceptable coverage, given as a non-negative `integer` value. All methylation measurements with lower coverage than this threshold are set to `NA`. If this parameter is `0`, calling this method has no effect.

Value

List of three elements:

`"dataset.before"` Copy of `rnb.set`.

`"dataset"` The (possibly) modified dataset after retaining sites on autosomes only.

`"mask"` A logical matrix of dimension `meth(rnb.set,type="sites")` indicating which methylation values have been masked

Author(s)

Fabian Mueller

`rnb.execute.lump` *Leukocytes unmethylation for purity*

Description

Implementation of the LUMP (Leukocytes UnMethylation for Purity) algorithm for purity estimation on methylation datasets.

Usage

```
rnb.execute.lump(dataset)
```

Arguments

`dataset`	Methylation dataset to study, provided as an object of type inheriting `RnBSet`.

Details

The LUMP algorithm is developed by Dvir Aran, Marina Sirota and Atul J. Buttea.

Value

Purity esimates provided as a `vector` of values in the range `[0, 1]`. The attribute `"sites"` contains the number of sites used in estimating the immune cell proportions. In case the dataset does not contain measurements for any of the sites on which LUMP focuses, the return values is `NULL`.

Author(s)

Yassen Assenov

`rnb.execute.na.removal`

rnb.execute.na.removal

Description

Removes all probes with missing value (if such exists) from the given dataset.

Usage

```
rnb.execute.na.removal(
  rnb.set,
  threshold = rnb.getOption("filtering.missing.value.quantile")
)
```

Arguments

`rnb.set`	Methylation dataset as an object of type inheriting `RnBSet`.
`threshold`	Maximum quantile of `NA`s allowed per site. This must be a value between 0 and 1.

Value

List of four or five elements:

`"dataset.before"` Copy of `rnb.set`.

`"dataset"` The (possibly modified) dataset after performing the missing value removal.

`"filtered"` `integer` vector storing the indices (in beta matrix of the unfiltered dataset) of all removed sites.

`"threshold"` Copy of `threshold`.

`"naCounts"` Vector storing the number of NAs per site

Author(s)

Yassen Assenov

Examples

```
library(RnBeads.hg19)
data(small.example.object)
rnb.set.filtered <- rnb.execute.na.removal(rnb.set.example, 0)$dataset
identical(meth(rnb.set.example), meth(rnb.set.filtered)) # TRUE
```

`rnb.execute.normalization`

rnb.execute.normalization

Description

Performs normalization of the provided HumanMethylation450 data set.

Usage

```
rnb.execute.normalization(
  object,
  method = rnb.getOption("normalization.method"),
  bgcorr.method = rnb.getOption("normalization.background.method"),
  verbose = TRUE
)
```

Arguments

`object`	Methylation dataset as an object of type `MethyLumiSet` or `RnBSet`.
`method`	Normalization method, must be one of `"none"`, `"illumina"`, `"swan"`, `"minfi.funnorm"`, `"bmiq"`, or `wm.*` where `*` stands for one of the methods implemented in **wateRmelon** package. Note that the execution of methods SWAN and minfi.funnorm requires packages **minfi** and **IlluminaHumanMethylation450kmanifest**. The BMIQ method requires the package **RPMM**. The `wm.*` methods naturally require **wateRmelon**.
`bgcorr.method`	Character singleton specifying which background subtraction should be used. Only methods impemented in the **methylumi** package are supported at the moment, namely `methylumi.noob`, `methylumi.goob` and `methylumi.doob`. See Triche et al. for detailed description of the methods.
`verbose`	flag specifying whether diagnostic output should be written to the console or to the RnBeads logger in case the latter is initialized

Value

Normalized dataset as an object of type `RnBeadSet`.

Author(s)

Pavlo Lutsik

References

1. Triche, Timothy J., Jr., Weisenberger, Daniel J., Van Den Berg, David, Laird, Peter W. and Siegmund, Kimberly D. (2013) Low-level processing of Illumina Infinium DNA Methylation BeadArrays. Nucleic Acids Research 41(7):e90-e90.

Examples

```
library(RnBeads.hg19)
data(small.example.object)
logger.start(fname=NA)
rnb.set.norm<-rnb.execute.normalization(rnb.set.example, method="illumina", bgcorr.method="none")
```

`rnb.execute.pOOBAH` *rnb.execute.pOOBAH*

Description

Probe signal intensities are masked based on their out-of-band signal intensities to counter hybridization failure.

Usage

```
rnb.execute.pOOBAH(
  raw.set,
  anno.table = NULL,
  pval.thresh = 0.05,
  verbose = FALSE
)
```

Arguments

`raw.set`	Methylation dataset as an instance of `RnBeadRawSet`.
`anno.table`	Annotation for `raw.set`.
`pval.thresh`	Computed detection p-values above this threshold are masked. Default value is 0.05.
`verbose`	If set to true, a short information is printed on how many probes are masked by the method.

Details

rnb.execute.pOOBAH is used to apply the method *pOOBAH* (P-value with OOB probes for Array Hybridization), which was conceived by Zhou, Triche, Laird and Shen to mask probes associated with hybridization failures. pOOBAH has been implemented in the R-package *"sesame"*, a dependency needed for this function (see Zhou et al, 2018 and the respective Bioconductor/github pages). *pOOBAH* computes the detection p-values by constructing 2 empirical cumulative density functions (eCDFs) based on the out-of-band signal intensities of the red and the green channel, respectively, to detect hybridization failures. The (in-band) green and red channel signal intensities of the probes are passed to the eCDFs and the probes with a p-value higher than the given threshold (`pval.thresh`) are masked, as they are considered background. *pOOBAH* is applied *separately to each sample*. Hybridization failures might occur due to somatic or germline deletions. In addition, unreliable low-intensity probes might also be masked.

Value

Returns a modified `RnBeadRawSet`, in which signal intensities are masked, if their computed p-value was greater than `pval.thresh`. Note, in datasets with several samples, signal intensities of a specific probe might be masked in sample A, but not in sample B, as *pOOBAH* is applied separately to each sample. For example: the signal intensities of probe cg24488772 might be masked in sample 1, but not in sample 12.

Author(s)

pOOBAH method: Wanding Zhou. Adapted by Nathan Steenbuck.

Examples

```
library(RnBeads.hg19)
data(small.example.object)
```

`rnb.execute.quality` *rnb.execute.quality*

Description

Performs quality control calculations on the loaded DNA methylation data set.

Usage

```
rnb.execute.quality(
  object,
  type = "sites",
  qc.coverage.plots = rnb.getOption("qc.coverage.plots"),
  verbose = TRUE
)
```

Arguments

`object`	Methylation dataset as an object of class `RnBeadSet`, `RnBeadRawSet` or `RnBiseqSet`.
`type`	`character` vector of length 1 giving the type of genomic regions for which the quality control information is summarized.
`qc.coverage.plots`	Flag indicating if sequencing coverage information is summarized and returned. This parameter is considered only when `object` is of type `RnBiseqSet`.
`verbose`	Flag specifying whether diagnostic output should be written to the console or to the RnBeads logger in case the latter is initialized.

Details

Currently, summarizing coverage for `RnBiseqSet` object is the only available function.

Value

`RnBeadSet` object with imputed quality control information

Author(s)

Pavlo Lutsik

`rnb.execute.segmentation` *rnb.execute.segmentation*

Description

This function computes methylation segmentation by MethylSeekR into PMDs, UMRs/LMRs, and HMDs. It is recommened to only execute this function on WGBS data (with coverage >=10 according to the developer's recommendation), but could also be used with RRBS_HaeIII without guarantee and the results should be interpreted carefully.

Usage

```
rnb.execute.segmentation(
  rnb.set,
  sample.name,
  meth.level = 0.5,
  fdr = 5,
  min.cover = 5,
  n.cores = 1,
  chr.sel = "chr2",
  plot.path = getwd(),
  temp.dir = tempdir()
)
```

Arguments

`rnb.set`	An object of type `RnBiseqSet-class` containing methylation and coverage information.
`sample.name`	The sample for which segmentation is to be executed. Segemntation can only be exectued for each sample individually.
`meth.level`	Methylation cutoff to be used in UMR/LMR computation
`fdr`	False discovery rate cutoff to be used in percent
`min.cover`	The coverage threshold
`n.cores`	The number of cores available for analysis
`chr.sel`	Selected chromosome for model training in "chrAB" format. Defaults to "chr2".
`plot.path`	Location on disk on which diagnostic plots are to be stored. Defaults to the working directory.
`temp.dir`	The temporary directory. Defaults to the R temporary directory.

Details

For further descriptions on the methods, see `MethylSeekR`-documentation. The new annotations can be accessed via `rnb.get.annotation("[PMDs,UMRs,LMRs,HMDs]_[sample.name]")`.

Value

The input RnBSet object with segementation added as an additional region type. Furthermore, three new annotations are set globally containing segmentation into PMDs, UMRs/LMRs, and HMDs for the sample that was specified.

Author(s)

Michael Scherer, based on a script by Abdulrahman Salhab

References

1. Burger, Lukas, Gaidatzis, Dimos, Schuebeler, Dirk, and Stadler, Michael B. (2013) Identification of active regulatory regions from DNA methylation data. Nucleic Acids Research 41(16):e155.

`rnb.execute.sex.prediction` *rnb.execute.sex.prediction*

Description

Infers the sex of every sample in the given dataset, based on average signal intensity values on the autosomes and the sex chromosomes.

Usage

```
rnb.execute.sex.prediction(rnb.set)
```

Arguments

`rnb.set`	Methylation dataset as an object of type `RnBeadRawSet`.

Value

The possibly modified dataset. If sex could be predicted, the sample annotation table is enriched with two more columns - `"Predicted Male Probability"` and `"Predicted Sex"`.

Author(s)

Yassen Assenov

Examples

```
library(RnBeads.hg19)
data(small.example.object)
rnb.set.example <- rnb.execute.sex.prediction(rnb.set.example)
table(rnb.set.example[, "Predicted Sex"])
```

`rnb.execute.sex.removal`

rnb.execute.sex.removal

Description

Removes all sites in sex chromosomes from the given dataset.

Usage

```
rnb.execute.sex.removal(rnb.set)
```

Arguments

`rnb.set`	Methylation dataset as an object of type inheriting `RnBSet`.

Value

List of three elements:

`"dataset.before"` Copy of `rnb.set`.

`"dataset"` The (possibly) modified dataset after retaining sites on autosomes only.

`"filtered"` `integer` vector storing the indices (in beta matrix of the unfiltered dataset) of all removed probes.

Author(s)

Yassen Assenov

Examples

```
library(RnBeads.hg19)
data(small.example.object)
rnb.set.filtered <- rnb.execute.sex.removal(rnb.set.example)$dataset
identical(meth(rnb.set.example), meth(rnb.set.filtered)) # FALSE
```

`rnb.execute.snp.removal`

rnb.execute.snp.removal

Description

Removes all probes overlapping with single nucleotide polymorphisms (SNPs) from the given dataset.

Usage

```
rnb.execute.snp.removal(rnb.set, snp = rnb.getOption("filtering.snp"))
```

Arguments

`rnb.set`	Methylation dataset as an object of type inheriting `RnBSet`.
`snp`	Criterion for the removal of sites or probes based on overlap with SNPs. Possible values are `"no"`, `"3"`, `"5"`, `"any"` or `"yes"`. See the documentation of `rnb.options` for a detailed explanation of the procedures these values encode.

Value

`list` of four elements:

`"dataset.before"` Copy of `rnb.set`.

`"dataset"` The (possibly) modified dataset object after removing probes that overlap with SNPs.

`"filtered"` `integer` vector storing the indices (in beta matrix of the unfiltered dataset) of all removed sites or probes.

`"snp"` The value of the `snp` parameter.

Author(s)

Yassen Assenov

Examples

```
library(RnBeads.hg19)
data(small.example.object)
rnb.set.filtered <- rnb.execute.snp.removal(rnb.set.example, "any")$dataset
identical(meth(rnb.set.example), meth(rnb.set.filtered)) # FALSE
```

`rnb.execute.sva` *rnb.execute.sva*

Description

Conduct Surrogate Variable Analysis (SVA) on the beta values of an RnBSet for given target variables

Usage

```
rnb.execute.sva(
  rnb.set,
  cmp.cols = rnb.getOption("inference.targets.sva"),
  columns.adj = rnb.getOption("covariate.adjustment.columns"),
  assoc = TRUE,
  numSVmethod = rnb.getOption("inference.sva.num.method")
)
```

Arguments

`rnb.set`	The `RnBSet` object on which the SVA should be conducted
`cmp.cols`	a vector of sample annotation column names which will be the targets of the SVA.
`columns.adj`	Column names in the table of phenotypic information to be used for confounder adjustment.
`assoc`	a flag indicating whether association information with principal components and other sample annotation should be returned
`numSVmethod`	method to estimate the number of surrogate variables. Passed to `sva`.

Value

An object of class `SvaResult`: basically a list containing the following elements:

`num.components` a vector storing the number of detected SVs for each target variable

`sva.performed` a vector storing whether SVA was performed on a target variable and whether more than 0 SVs were found

`targets` a vector storing the names of the target variables

`components` a list storing for each target variable a matrox containing the sample-wise SVs as rows

`assoc` a special object containing association information of SVs with principal components and sample annotations typically only used `rnb.section.sva`.

Author(s)

Fabian Mueller

Examples

```
library(RnBeads.hg19)
data(small.example.object)
logger.start(fname=NA)
sva.obj <- rnb.execute.sva(rnb.set.example,c("Sample_Group","Treatment"),numSVmethod="be")
sva.obj$sva.performed
sva.obj$num.components
rnb.set.mod <- set.covariates.sva(rnb.set.example, sva.obj)
has.covariates.sva(rnb.set.example,"Sample_Group")
has.covariates.sva(rnb.set.mod,"Sample_Group")
has.covariates.sva(rnb.set.mod,"Treatment")
```

`rnb.execute.tnt` *rnb.execute.tnt*

Description

export RnBSet to various output data formats

Usage

```
rnb.execute.tnt(
  rnb.set,
  out.dir,
  exp.bed = rnb.getOption("export.to.bed"),
  exp.trackhub = rnb.getOption("export.to.trackhub"),
  region.types = rnb.getOption("export.types"),
  ...
)
```

Arguments

`rnb.set`	`RnBSet` object
`out.dir`	output directory.
`exp.bed`	A character vector indicating which data types should be exported to UCSC. Possible values in the vector are `bigBed` and `bigWig`. If `NULL`, UCSC export is disabled
`exp.trackhub`	file types which should be exported to a trackhub structure.
`region.types`	a character vector indicating region types to be exported
`...`	Arguments passed to `rnb.export.to.trackhub`

Value

a list containing information on the export

Author(s)

Fabian Mueller

Examples

```
library(RnBeads.hg19)
data(small.example.object)
logger.start(fname=NA)
rnb.execute.tnt(rnb.set.example,tempdir())
```

`rnb.execute.training` *rnb.execute.training*

Description

Trains a new age predictor on the specified data set and writes it to the given path. Elastic net regression is to fit the input ages to the methylation values .

Usage

```
rnb.execute.training(object, path = "", alpha = 0.8)
```

Arguments

`object`	a `RnBSet` object on which a new predictor should be created
`path`	path to which the predictor should be written out
`alpha`	alpha parameter used in the elastic net regression

Author(s)

Michael Scherer

`rnb.execute.variability.removal`

rnb.execute.variability.removal

Description

Removes all sites or probes with low variability from the given dataset.

Usage

```
rnb.execute.variability.removal(
  rnb.set,
  min.deviation = rnb.getOption("filtering.deviation.threshold")
)
```

Arguments

`rnb.set`	Methylation dataset as an object of type inheriting `RnBSet`.
`min.deviation`	Threshold for standard deviation per site. This must be a scalar between 0 and 1. All sites, for which the standard deviation of methylation values (for all samples in `rnb.set`) is lower than this threshold, will be filtered out.

Value

List of four elements:

`"dataset.before"` Copy of `rnb.set`.

`"dataset"` The (possibly modified) dataset after removing sites with low variability.

`"filtered"` `integer` vector storing the indices (in beta matrix of the unfiltered dataset) of all removed sites.

`"threshold"` The value of the given parameter `min.deviation`.

Author(s)

Yassen Assenov

Examples

```
library(RnBeads.hg19)
data(small.example.object)
rnb.set.filtered <- rnb.execute.variability.removal(rnb.set.example, 0.01)
```

`rnb.export.all.annotation` *rnb.export.all.annotation*

Description

Wrapper for exporting all annotation sets

Usage

```
rnb.export.all.annotation(
  out.dir,
  types = c("CpG", rnb.region.types(assembly)),
  assembly = "hg19",
  format = "bed"
)
```

Arguments

`out.dir`	The directory to write the files to
`types`	One-element `character` vector giving the name of the region annotation.
`assembly`	Genome assembly of interest. See `rnb.get.assemblies` for the list of supported genomes.
`format`	output format. currently only "bed" is supported.

Value

`TRUE`, invisibly.

Author(s)

Fabian Mueller

Examples

```
logger.start(fname=NA)
rnb.export.all.annotation(tempdir(),c("genes","promoters"))
```

`rnb.export.annotation` *rnb.export.annotation*

Description

Export the annotation to a defined format (currently only bed is supported

Usage

```
rnb.export.annotation(fname, type, assembly = "hg19", format = "bed")
```

Arguments

`fname`	One-element `character` vector giving the name of the file to contain the annotation data. If this file already exists, it will be overwritten.
`type`	One-element `character` vector giving the name of the region annotation.
`assembly`	Genome assembly of interest. See `rnb.get.assemblies` for the list of supported genomes.
`format`	Output format. currently only `"bed"` is supported.

Value

`TRUE`, invisibly.

Author(s)

Fabian Mueller

Examples

```
rnb.export.annotation(tempfile(pattern="promoters",fileext=".bed"),"promoters")
```

`rnb.export.to.ewasher` *rnb.export.to.ewasher*

Description

Data exported to a format compatible with the FaST-LMM-EWASher tool for cell-mixture adjustment. see Zou, J., et al., Nature Methods, 2014 for further details on the tool.

Usage

```
rnb.export.to.ewasher(rnb.set, out.dir, reg.type = "sites", ...)
```

Arguments

`rnb.set`	Object of class `RnBSet`
`out.dir`	output directory. If not existing, it will be created and all exported files will be placed here. If existing, this functions results in an error.
`reg.type`	region type to be exported
`...`	passed on to `get.comparison.info`

Value

a list containing information on the export

Author(s)

Fabian Mueller

Examples

```
library(RnBeads.hg19)
data(small.example.object)
logger.start(fname=NA)
rnb.export.to.ewasher(rnb.set.example,tempfile(pattern="forEwasher"))
```

`rnb.export.to.trackhub` *rnb.export.to.trackhub*

Description

convert an `RnBSet` object to a UCSC-style track hub.

Usage

```
rnb.export.to.trackhub(
  rnb.set,
  out.dir,
  reg.type = "sites",
  data.type = "bigBed",
  ...
)
```

Arguments

`rnb.set`	Object of class `RnBSet`
`out.dir`	output directory. If not existing, it will be created. otherwise files in that directory are overwritten.
`reg.type`	region type to be converted
`data.type`	either "bigBed" or "bigWig"
`...`	parameters passed on to the track hub generating procedure

Details

During execution the RnBSet is converted to bed files. If the operating system is supported (currently Unix and MacOS only) these are automatically converted to bigBed files. If your operating system is not supported, you need to create them manually (see the UCSC Genome Browser documentation for details). For details on UCSC track hubs see the UCSC tracks help page.

Value

a list containing information on the export

Author(s)

Fabian Mueller

Examples

```
library(RnBeads.hg19)
data(small.example.object)
logger.start(fname=NA)
rnb.export.to.trackhub(rnb.set.example,tempdir())
```

`rnb.find.relative.site.coord` *rnb.find.relative.site.coord*

Description

given a region types, assigns sites to regions and determines relative positions of sites in the assigned region

Usage

```
rnb.find.relative.site.coord(rnb.set, region.type, extend.by = 0.33)
```

Arguments

`rnb.set`	RnBSet object
`region.type`	Region type for which the coordinates are computed
`extend.by`	A number between 0 and 1 specifying the percentage by which a region is extended in order to capture methylation information before region start and after region end

Value

a data frame containing the site index, the assigned region index and the relative coordinate The relative coordinate is 0 if the site's coordinate is identical to the region start coordinate and 1 if identical to the regions end coordinate and scaled inbetween. Coordinates can be less than 0 or larger than 1 if a site is in the upstream or downstream flanking region respectively

Author(s)

Fabian Mueller

`rnb.get.annotation` *rnb.get.annotation*

Description

Extracts the requested annotation for the given genome.

Usage

```
rnb.get.annotation(type = "CpG", assembly = "hg19")
```

Arguments

`type`	Name of annotation.
`assembly`	Genome assembly of interest. See `rnb.get.assemblies` for the list of supported genomes.

Details

When the returned value is of type `GRangesList`, it defines the genomic positions of the requested sites, probes or regions. Identifiers, if present, can be obtained using the `names` method. Strand information is also included when applicable. Any additional annotation is stored as metadata in the respective `GRanges` objects.

Value

Probe, site or region annotation table. If the specified type refers to control probes, the returned value is a `data.frame` listing all respective control probes. Otherwise, this function returns an object of type `GRangesList` - a list of consistent `GRanges` objects, one per chromosome.

Author(s)

Fabian Mueller

See Also

`rnb.set.annotation` for adding annotation; `rnb.region.types` for all loaded region types in a genome assembly

Examples

```
rnb.get.annotation("promoters")
```

`rnb.get.assemblies` *rnb.get.assemblies*

Description

Gets the supported genome assemblies.

Usage

```
rnb.get.assemblies()
```

Value

All supported genome assemblies in the form of a `character` vector. These are `"hg19"`, `"mm10"`, `"mm9"` and `"rn5"`.

Author(s)

Yassen Assenov

Examples

```
"hg19" %in% rnb.get.assemblies()
```

`rnb.get.chromosomes` *rnb.get.chromosomes*

Description

Gets the chromosome names supported for the specified assembly.

Usage

```
rnb.get.chromosomes(assembly = "hg19")
```

Arguments

`assembly`	Genome assembly of interest. See `rnb.get.assemblies` for the list of supported genomes.

Value

`character` vector of supported chromosomes for the specified genome assembly. The elements of the vector follow the Ensembl convention (`"1"`, `"2"`, ...), and the names of this vector - the convention of the UCSC Genome Browser (`"chr1"`, `"chr2"`, ...).

Author(s)

Pavlo Lutsik

Examples

```
"chrX" %in% names(rnb.get.chromosomes())
```

`rnb.get.directory` *rnb.get.directory*

Description

Gets the location of the given report-specific directory.

Usage

```
rnb.get.directory(
  report,
  dir = c("data", "images", "images-high", "pdfs"),
  absolute = FALSE
)
```

Arguments

`report`	Report of interest.
`dir`	Type of directory to get. Must be one of `"data"`, `"images"`, `"images-high"` or `"pdfs"`.
`absolute`	Flag indicating if the absolute path of the directory is to be returned. If this is `FALSE`, the directory name is returned relative to the report's HTML file location.

Value

Path of the requested directory as a single-element `character` vector.

Author(s)

Yassen Assenov

See Also

`Report` for functions adding contents to an HTML report

Examples

```
report <- createReport("example.html", "Example", init.configuration = TRUE)
rnb.get.directory(report, "data")
```

`rnb.get.mapping` *rnb.get.mapping*

Description

Gets the mapping information used for a region type. These are structures used to map regions to the genomic loci (or Infinium probes) that target them.

Usage

```
rnb.get.mapping(region.type, target.type, assembly = "hg19")
```

Arguments

`region.type`	Region type. The built-in types are `"cpgislands"`, `"genes"`, `"promoters"` and `"tiling"`.
`target.type`	Target type for sites.
`assembly`	Genome assembly of interest. See `rnb.get.assemblies` for the list of supported genomes.

Value

`list` of mapping structures, one per chromosome. Every mapping structure is an object of type `IRanges` and stores the range of indices of all sites contained in the respective region. Regions that do not contain sites are left out of the mapping.

Author(s)

Yassen Assenov

Examples

```
promoters2probes <- rnb.get.mapping("promoters", "probes450")
promoters2probes[["chr21"]]
```

`rnb.get.reference` *rnb.get.reference*

Description

Creates a string that points to the given reference item in the specified report.

Usage

```
rnb.get.reference(report, txt)
```

Arguments

`report`	Report that contains the reference to be cited.
`txt`	Text of the reference in the form of a non-empty `character` vector. This reference must already added to the report.

Value

Citation of the reference item (including a link) in the form of a one-element `character` vector. If the specified reference item is not found in the report, this method returns an empty string.

Author(s)

Yassen Assenov

See Also

`rnb.add.reference` for adding a reference item to a report; `Report` for other functions adding contents to an HTML report

Examples

```
report <- createReport("example.html", "Example", init.configuration = TRUE)
txt.reference <- c("Bird A. ", "<i>Nucleic Acids Res.</i> <b>8</b> (1980)")
report <- rnb.add.reference(report, txt.reference)
txt <- c("This was shown in ", rnb.get.reference(report, txt.reference), ".")
rnb.add.paragraph(report, txt)
```

`rnb.get.reliability.matrix`
rnb.get.reliability.matrix

Description

Gets a matrix of reliability indications for every measurement in the given dataset.

Usage

```
rnb.get.reliability.matrix(rnb.set, row.names = FALSE)
```

Arguments

`rnb.set`	Methylation dataset as an object of type inheriting `RnBSet`.
`row.names`	Flag indicating of row names are to be generated in the result.

Value

`logical` matrix in which every row corresponds to a CpG site or probe and every column - to a patient. If the dataset does not contain coverage or detection p-value information, the returned value is `NULL`.

Author(s)

Yassen Assenov

Examples

```
library(RnBeads.hg19)
data(small.example.object)
rnb.options(identifiers.column = "Sample_ID")
str(rnb.get.reliability.matrix(rnb.set.example))
```

`rnb.infinium.control.targets` *rnb.infinium.control.targets*

Description

Extracts all control probe types in the HumanMethylation450 assay.

Usage

```
rnb.infinium.control.targets(target = "probes450")
```

Arguments

`target`	A singleton of type `character`, specifying the microarray platform. `"probesEPICv2"`,`"probesEPIC"` and `"probes27"` correspond to MethylationEPICv2, Methylation EPIC, HumanMethylation450, and HumanMethylation27 microarrays respectively.

Value

`character` vector of control targets.

Author(s)

Pavlo Lutsik

Examples

```
"NEGATIVE" %in% rnb.infinium.control.targets()
```

`rnb.initialize.reports` *rnb.initialize.reports*

Description

Creates a new directory to host HTML reports and copies the shared configuration files.

Usage

```
rnb.initialize.reports(dir.reports, dir.configuration = "configuration")
```

Arguments

`dir.reports`	Directory to host report files. This must be a `character` of length one that specifies a non-existent path, as this methods attempts to create it.
`dir.configuration`	Subdirectory to host configuration files shared by the reports. This must be a `character` of length one that gives location as a path relative to `dir.reports`. Also, strong restrictions apply to the path name. See the description of the `createReport` function for more details. This method creates the directory and copies configuration files that define cascading style sheet (CSS) definitions and Javascript functions used by the HTML reports.

Value

`TRUE` if the report directory was successfully created and the configuration files were copied to the specified location; `FALSE` otherwise.

Author(s)

Yassen Assenov

See Also

`createReport` for initializing an HTML report

Examples

```
dir.reports <- "~/infinium_studies/cancer_study/reports"
if (!rnb.initialize.reports(dir.reports)) {
    cat("ERROR: Could not initialize configuration in ", dir.reports, "\n", sep = "")
}
```

`rnb.is.option` *rnb.is.option*

Description

Checks if the specified text is an option name.

Usage

```
rnb.is.option(txt)
```

Arguments

`txt`	Potential option name. This should be a one-element `character` vector.

Value

`TRUE` if the specified parameter is a valid analysis option name; `FALSE` otherwise.

Author(s)

Yassen Assenov

See Also

`rnb.options` for getting and setting option values

Examples

```
rnb.is.option("logging") # TRUE
rnb.is.option("Logging") # FALSE
```

`rnb.load.annotation` *rnb.load.annotation*

Description

Loads a previously saved custom region annotation from a binary (RData) file.

Usage

```
rnb.load.annotation(fname, type)
```

Arguments

`fname`	One-element `character` vector giving the name of the file that contains the annotation data.
`type`	One-element `character` vector giving the name of the region annotation. If this annotation is already available, it will be overwritten for the current session.

Details

If the region annotation cannot be loaded from the specified location, this function exits with an error message in the form `"unable to load object from ..."`. This could happen, for example, when `fname` does not refer to a valid RData file, or the file cannot be accessed due to security restrictions.

If the file is loaded in the current session, but no annotation was added, the function returns invisibly one of the following short failure messages:

`"invalid format"` The RData file does not store exactly the following three objects - `assembly`, `regions`, and `mapping`, or they are not of the expected type.

`"unsupported assembly"` The specified assembly is unknown.

`"invalid format of regions"` The specified region annotation table is invalid.

`"invalid format of mappings"` The specified region mapping tables are invalid.

Value

Invisibly, `TRUE` if the annotation was loaded successfully; an error message if the objects in the given file do not encode an annotation.

Author(s)

Yassen Assenov

See Also

`rnb.save.annotation` for saving annotation to a binary file; `rnb.set.annotation` for loading an annotation from a BED file.

`rnb.load.annotation.from.db`

rnb.load.annotation.from.db

Description

Loads a previously region annotation from the RnBeads resource database

Usage

```
rnb.load.annotation.from.db(types, assembly = "hg19")
```

Arguments

`types`	One-element `character` vector giving the name of the region annotation. If this annotation is already available, it will be overwritten for the current session.
`assembly`	Genome assembly of interest. See `rnb.get.assemblies` for the list of supported genomes.

Details

This function checks whether a region annotation is present in the RnBeads resources, downloads the corresponding annotation file(s) from the and then runs `rnb.load.annotation` to import the annotation.

Value

Invisibly, `TRUE` if the annotation was loaded successfully; an error message if the objects in the given file do not encode an annotation.

Author(s)

Fabian Mueller

See Also

`rnb.load.annotation` for loading annotation from a binary file

Examples

```
rnb.region.types()
rnb.load.annotation.from.db(c("tiling1kb", "dynamicMethZiller2013"))
rnb.region.types()
```

`rnb.load.sitelist` *rnb.load.sitelist*

Description

Loads a list of probe or site identifiers. This function is used in the preprocessing module for loading a whitelist and/or a blacklist of identifiers.

Usage

```
rnb.load.sitelist(fname, verbose = FALSE)
```

Arguments

`fname`	File listing the identifiers, one per line.
`verbose`	Flag indicating if messages are to be printed. If the values is `TRUE` and a logger is initialized, this function adds a message to the log.

Value

The loaded list of identifiers, or `NULL` if `fname` could not be open.

Author(s)

Yassen Assenov

See Also

`logger.start` for initializing a logger

`rnb.message.plot` *rnb.message.plot*

Description

Creates a plot, using **ggplot2**, with a single text message.

Usage

```
rnb.message.plot(txt)
```

Arguments

`txt`	Text to be plotted.

Value

The newly initialized `ggplot` instance.

Author(s)

Yassen Assenov

Examples

```
x11(width = 5, height = 5)
rnb.message.plot("Missing data")
```

`rnb.mval2beta` *rnb.mval2beta*

Description

Transforms M values to beta values.

Usage

```
rnb.mval2beta(mvals)
```

Arguments

`mvals`	`numeric` vector or matrix of M values to be transformed.

Value

The calculated beta values.

Author(s)

Pavlo Lutsik

Examples

```
library(RnBeads.hg19)
data(small.example.object)
logger.start(fname=NA)
mvals <- rnb.beta2mval(meth(rnb.set.example))
bvals <- rnb.mval2beta(mvals)
all((bvals-meth(rnb.set.example))<1e-10)
```

`rnb.options` *RnBeads Options*

Description

Allows the user to set and examine a variety of **RnBeads** global options. They affect the way in which the package computes and displays its results.

Usage

```
rnb.options(...)

rnb.getOption(x)
```

Arguments

`...`	Option names as `character`s, or new option values given in the form `name = value`.
`x`	Option name in the form of a `character` vector of length 1.

Details

Invoking `rnb.options()` with no arguments returns a list with the current values of the options. To access the value of a single option, one should use, e.g., `rnb.getOption("filtering.greedycut")`, rather than `rnb.options("filtering.greedycut")` which is a *list* of length one. Also, only a limited set of options is available (see below). Attempting to get or set the value of a non-existing option results in an error.

Value

For `rnb.getOption`, the current value for `x`. For `rnb.options()`, a list of all **RnBeads** options and their current values. If option names are given, a list of all requested options and their values. If option values are set, `rnb.options` returns the previous values of the modified options, invisibly.

Options used in RnBeads

`analysis.name` = `NULL` One-element `character` vector storing a short title of the analysis. If specified, this name appears at the page title of every report.

`logging` = `TRUE` Flag indicating if logging functionality is enabled in the automatic runs of the pipeline.

`email` = `NULL` Email address associated with the analyses.

`assembly` = `"hg19"` Genome assembly to be used. Currently only important for bisulfite mode. The supported genomes returned by the function `rnb.get.assemblies`.

`analyze.sites` = `TRUE` Flag indicating if analysis on site or probe level is to be conducted. Note that the preprocessing module always operates on the site level (only), regardless of the value of this option.

`preanalysis.scriptno default` Path to an R script that is executed before every module. This options is useful if e.g., if you specify custom regions that have to be present in a cluster enviroment.

`region.types` = `NULL` Region types to carry out analysis on, in the form of a `character` vector. `NULL` (default value) signifies that all available region annotations (as returned by `rnb.region.types`) are summarized upon loading and normalization, and the other modules analyze all regions summarized in the dataset. If this option is set to an empty vector, analysis on the region level is skipped.

`region.aggregation` = `"mean"` Aggregation function to apply when calculating the methylation value for a region based on the values of the CpGs associated with that region. Accepted values for this function are `"min"`, `"max"`, `"mean"` (default), `"median"`, `"sum"`, `"coverage.weighted"`. The last method is applicable only for sequencing-based methylation datasets. It computes the weighted average of the values of the associated CpGs, whereby weights are calculated based on the coverages of the respective sites.

`region.subsegments` = `0` If a number larger than 1 is specified, **RnBeads** will subdivide each region specified in the `region.types` option into subsegments containing on average `region.subsegments` sites per subsegment. This is done by clustering the sites within each regions according to their genomic coordinates. These subsegments are then used for subsequent analysis. Use cautiously as this will significantly increase the runtime of the pipeline.

`region.subsegments.types` = `NULL` The region types to which subsegmentation will be applied. Defaults to `region.types` when set to `NULL`.

`identifiers.column` = `NULL` Column name or index in the table of phenotypic information to be used when plotting sample identifiers. If this option is `NULL`, it points to a non-existing column or a column that does not list IDs, the default identifiers are used. These are the row names of the sample phenotype table (and the column names of the beta value matrix).

`colors.category` = `c("#1B9E77","#D95F02",...)` `character` vector of length 2 or more giving the color scheme for displaying categorical trait values in plots. RnBeads denotes missing values (`NA`) by grey, therefore, it is not recommended to include shades of grey in this vector. The default value of this option is the result of the `"Dark2"` palette of *RColorBrewer* with 8 values.

`colors.gradient` = `c("#132B43","#56B1F7")` `character` vector of length 2 or more giving the color scheme for displaying continuous (gradient) trait values in plots. **RnBeads** interpolates between the color values.

`min.group.size` = `2` Minimum number of samples each subgroup defined by a trait, in order for this trait to be considered in the methylation profiles and in the differential methylation modules. This must be a positive `integer`.

`max.group.count` = `NULL` Maximum number of subgroups defined by a trait, in order for this trait to be considered in the methylation profiles and in the differential methylation modules. This must be an `integer` of value `2` or more. As a special case, a value of `NULL` (default) indicates that the maximum number of subgroups is the number of samples in an analysis minus `1`, i.e. traits with all unique values will be ignored.

`replicate.id.column` = `NULL` Column name in the sample annotation table that indicates sample replicates. Replicates are expected to contain the same value. Samples without replicates should contain unique or missing values. If this option is `NULL` (default), replicate handling is disabled.

`gz.large.files` = `FALSE` Flag indicating whether large output files should be compressed (in `.gz` format).

`import` = `TRUE` Flag controlling whether data import report should be generated. This option be set to `FALSE` only when the provided data source is an object of type RnBSet, i.e. the data has been previously loaded by **RnBeads**.

`import.default.data.type` = `"infinium.idat.dir"` Type of data assumed to be supplied by default (Infinium 450k microarray). For sequencing data set this to `bs.bed.dir` and save the options. See `rnb.execute.import` for further details.

`import.table.separator` = `","` Separator used in the plain text data tables. See `rnb.execute.import` for details.

`import.bed.style` = `"bismarkCov"` Preset for bed-like formats. `"BisSNP"`, `"Encode"`,`"EPP"`, `"bismarkCytosine"`, `"bismarkCov"` are currently supported. See the **RnBeads** vignette and the FAQ section on the website for more details.

`import.bed.columns` Column indices in the supplied BED file with DNA methylation information. These are represented by a named `integer` vector, in which the names are: `"chr"`, `"start"`, `"end"`, `"strand"`, `"meth"`, `"coverage"`, `"c"` and `"t"`. These names correspond the columns for chromosome, start position, end position, strand, methylation degree, read coverage, number of reads with C and number of reads with T, respectively. Methylation degree and/or read coverage, if not specified, are inferred from the values in the columns `"c"` and `"t"`. Further details and examples of BED files can be found in Section 4.1 of the RnBeads vignette.

`import.bed.frame.shift` = `1` Singleton of type `integer` specifying the frame shift between the coordinates in the input BED file and the corresponding genomic reference. This (`integer`) value is added to the coordinates from the BED file before matching the methylation sites to the annotated ones.

`import.bed.test` = `TRUE` Perform a small loading test, by reading 1000 rows from each BED file, after which normal loading is performed. See **RnBeads** vignette and the FAQ section on the website for more details.

`import.bed.test.only` = `FALSE` Perform only the small loading test, and skip loading all the data.

`import.skip.object.check` = `FALSE` Skip the check of the loaded RnBSet object after loading. Helps with keeping the memory profile down

`import.idat.platform` = `NULL` Character specifying the Infinium platform that is uses. Has to be one of `'probes27'`, `'probes450'`, `'probesEPIC'` or `'probesEPICv2'`. If `'auto'`, the platform is automatically detected from the IDAT file names.

`import.sex.prediction` = `TRUE` Flag indicating if sex prediction is to be performed. Sex prediction is supported for Infinium 450k, EPIC, EPICv2 and bisulfite sequencing datasets with signal intensity or coverage information. The value of this option is ignored for 27k datasets.

`qc` = `TRUE` Flag indicating if the quality control module is to be executed.

`qc.boxplots` = `TRUE` [Microarrays] Add boxplots for all types of quality control probes to the quality control report. The boxplots give signal distribution across samples.

`qc.barplots` = `TRUE` [Microarrays] Add barplots for each quality control probes to the quality control report.

`qc.negative.boxplot` = `TRUE` [Microarrays] Add boxplot of negative control probe intensities for all samples.

`qc.snp.heatmap` = `TRUE` [Microarrays] Flag indicating if a heatmap of the beta values for all SNP probes is to be generated.

`qc.snp.barplot` = `FALSE` [Microarrays] Add bar plots of the beta-values observed for each SNP-calling probe.

`qc.snp.boxplot` = `FALSE` [Microarrays] Add boxplot of beta-values for the SNP-calling probes.

`qc.snp.distances` = `TRUE` [Microarrays] Flag indicating if intersample distances based on the beta values of SNP probes are to be displayed. This can help identify genetically similar or identical samples.

`qc.snp.purity` = `FALSE` [Microarrays] Flag indicating if genetic purity should be estimated based on the beta values of SNP probes.

`qc.sample.batch.size` = `50` [Microarrays] Maximal number of samples included in a single quality control barplot and negative control boxplot.

`qc.coverage.plots` = `FALSE` [Bisulfite sequencing] Add genome-wide sequencing coverage plot for each sample.

`qc.coverage.threshold.plot` = `1:10` [Bisulfite sequencing] Values for coverage cutoffs to be shown in a coverage thresholds plot. This must be an `integer` vector of positive values. Setting this to an empty vector disables the coverage thresholds plot.

`qc.coverage.histograms` = `FALSE` [Bisulfite sequencing] Add sequencing coverage histogram for each sample.

`qc.coverage.violins` = `FALSE` [Bisulfite sequencing] Add sequencing coverage violin plot for each sample.

`qc.cnv` = `FALSE` [Microarrays] Add CNV estimation for each position in each sample.

`qc.cnv.refbased` = `TRUE` [Microarrays] Should CNV estimation be performed with a reference (twin study) or with the mean over the samples.

`preprocessing` = `TRUE` Flag controlling whether the data should be preprocessed (whether quality filtering and in case of Infinium microarray data normalization should be applied).

`normalization` = `NULL` Flag controlling whether the data should be normalized and normalization report generated. Setting this to `NULL` (default) enables this step for analysis on Infinium datasets, but disables it in case of sequencing-based datasets. Note that normalization is never applied in sequencing datasets; if this flag is enabled, it will lead to a warning message.

`normalization.method` = `"wm.dasen"` Normalization method to be applied, or `"none"`. Multiple normalization methods are supported: `"illumina"` - methylumi-implemented Illumina scaling normalization; `"swan"` - SWAN-normalization by Gordon et al., as implemented in minfi; `"bmiq"` - beta-mixture quantile normalization method by Teschendorff et al; as well as `"wm.dasen"` (default), `"wm.nasen"`, `"wm.betaqn"`, `"wm.naten"`, `"wm.nanet"`, `"wm.nanes"`, `"wm.danes"`, `"wm.danet"`, `"wm.danen"`, `"wm.daten1"`, `"wm.daten2"`, `"wm.tost"`, `"wm.fuks"` and `"wm.swan"` - all normalization methods implemented in the wateRmelon package. When setting this option to a specific algorithm, make sure its dedicated package is installed. `"scaling.internal"`, `"scalings"` - different scaling methods as implemented in the sesame R package.

`normalization.background.method` = `"none"` A character singleton specifying which background subtraction is to be performed during normalization. The following values are accepted: `"none"` (default), `"methylumi.noob"`, `"methylumi.goob"`, `"methylumi.lumi"`, `"enmix.oob"`, `"sesame.noob"` and `"sesame.noobsb"`.

`normalization.plot.shifts` = `TRUE` Flag indicating if the report on normalization should include plots of shifts (degrees of beta value correction).

`filtering.whitelist` = `NULL` Name of a file specifying site or probe identifiers to be whitelisted. Every line in this file must contain exactly one identifier. The whitelisted sites are always retained in the analysed datasets, even if filtering criteria or blacklisting requires their removal. For Infinium studies, the file must contain Infinium probe identifiers. For bisulfite sequencing studies, the file must contain CpG positions in the form "chromosome:coordinate" (1-based coordinate of the cytosine), e.g. `chr2:48607772`. Unknown identifiers are silently ignored.

`filtering.blacklist` = `NULL` Name of a file specifying site or probe identifiers to be blacklisted. Every line in this file must contain exactly one identifier. The blacklisted sites are removed from the analysed datasets as a first step in the preprocessing module. For Infinium studies, the file must contain Infinium probe identifiers. For bisulfite sequencing studies, the file must contain CpG positions in the form "chromosome:coordinate" (1-based coordinate of the cytosine), e.g. `chr2:48607772`. Unknown identifiers are silently ignored.

`filtering.context.removal` = `c("CC","CAG",...)` `character` vector giving the list of probe context types to be removed as a filtering step. Possible context values are `"CC"`, `"CG"`, `"CAG"`, `"CAH"`, `"CTG"`, `"CTH"` and `"Other"`. Probes in the second context measure CpG methylation; the last context denotes probes dedicated to SNP detection. Setting this option to `NULL` or an empty vector effectively disables the step of context-specific probe removal.

`filtering.snp` = `"any"` Removal of sites or probes based on overlap with SNPs. The accepted values for this option are:

`"no"` no SNP-based filtering;

`"3"` filter out a probe when the last 3 bases in its target sequence overlap with SNP;

`"5"` filter out a probe when the last 5 bases in its target sequence overlap with SNP;

`"any"` **or** `"yes"` filter out a CpG site or probe when any base in its target sequence overlaps with SNP.

Bisulfite sequencing datasets operate on sites instead of probes, therefore, the values `"3"` and `"5"` are treated as `"yes"`.

`filtering.cross.reactive` = `TRUE` Flag indicating if the removal of potentially cross-reactive probes should be performed as a filtering step in the preprocessing module. A probes whose sequence maps to multiple genomic locations (allowing up to 3 mismatches) is cross-reactive.

`filtering.greedycut` = `NULL` Flag indicating if the Greedycut procedure should be run as a filtering step in the preprocessing module. `NULL` (default) indicates that Greedycut will be run for array-based datasets, but not for sequencing-based datasets.

`filtering.greedycut.pvalue.threshold` = `0.05` Threshold for the detection p-value to be used in Greedycut. This is a value between 0 and 1. This option has effect only when `filtering.greedycut` is `TRUE`.

`filtering.greedycut.rc.ties` = `"row"` Indicator of what the behaviour of Greedycut should be in case of ties between the scores of rows (probes) and columns (samples). The value of this option must be one of `"row"`, `"column"` or `"any"`; the last one indicating random choice. This option has effect only when `filtering.greedycut` is `TRUE`.

`filtering.sex.chromosomes.removal` = `TRUE` Flag indicating if the removal of probes located on sex chromosomes should be performed as a filtering step.

`filtering.missing.value.quantile` = `0.5` Number between 0 and 1, indicating the fraction of allowed missing values per site. A site is filtered out when its methylation beta values are `NA`s in a larger fraction of samples than this threshold. Setting this option to 1 (default) retains all sites, and thus effectively disables the missing value filtering step in the preprocessing module. If this is set to 0, all sites that contain missing values are filtered out.

`filtering.coverage.threshold = 5` Threshold for minimal acceptable coverage. This must be a non-negative value. Setting this option to 0 (zero) effectively considers any known or unknown read coverage for sufficiently deep.

`filtering.low.coverage.masking = FALSE` Flag indicating whether methylation values for low coverage sites should be set to missing. In combination with `filtering.missing.value.quantile` this can lead to the removal of sites.

`filtering.high.coverage.outliers = FALSE` (Bisulfite sequencing mode) Flag indicating whether methylation sites with a coverage of more than 10 times the 95-percentile of coverage should be removed.

`filtering.deviation.threshold = 0` Threshold used to filter probes based on the variability of their assigned beta values. This must be a real value between 0 and 1, denoting minimum standard deviation of the beta values in one site across all samples. Any sites that have standard deviation lower than this threshold are filtered out. Note that sites with undetermined varibility, that is, sites for which there are no measurements (all beta values are `NA`s), are retained. Setting this option to 0 (default) disables filtering based on methylation variability.

`imputation.method = "none"` Character indicating which imputation method should be used to replace missing values. This option has to be one of the following values `"none"`, `"mean.cpgs"`, `"mean.samples"`, `"random"`, `"median.cpgs"`, `"median.samples"` or `"knn"`. Setting this option to `"none"` inactivates imputation (default).

`inference = FALSE` Flag indicating if the covariate inference analysis module is to be executed.

`inference.genome.methylation = "Genome-wide methylation"` Name of the column to add to the sample annotation, storing the genome-wide methylation level. If such a column already exists, its values will be overwritten. Setting this option to `NULL` or an empty `character` disables computing and adding genome-wide methylation levels.

`inference.targets.sva = character()` Column names in the sample annotation table for which surrogate variable analysis (SVA) should be conducted. An empty vector (default) means that SVA is skipped.

`inference.reference.methylome.column = character()` Column name in the sample annotation table giving the assignment of samples to reference methylomes. The target samples should have `NA` values in this column.

`inference.max.cell.type.markers = 50000` Number of most variable CpGs which are tested for association with the reference cell types. Setting this option to `NULL` forces the algorithm to use all available sites in the dataset, and may greatly increase the running time for cell type comoposition estimation.

`inference.top.cell.type.markers = 500` Number of top cell type markers used for determining cell type contributions to the target DNA methylation profiles using the projection method of Houseman et al.

`inference.sva.num.method = "leek"` Name of the method to be used for estimating the number of surrogate variables. must be either 'leek' or 'be', See `sva` function for details.

`inference.age.column = "age"` Name of the column in which the ages of the donors are annotated. This function can be of numeric, string or factor format.

`inference.age.prediction = TRUE` Flag indicating if the epigenetic age prediction within the inference module is to be executed.

`inference.age.prediction.training = FALSE` Flag indicating if a new predictor should be created based on the provided data set.

`inference.age.prediction.cv = FALSE` Flag indicating if predictive power of a predictor that was trained in that run of the age prediction should be assessed by cross-validation. This option only has an influence if `inference.age.prediction.training = TRUE`.

`inference.immune.cells` = `TRUE` Flag indicating if immune cell content estimation is to be performed. Immune cell content prediction is based on the LUMP algorithm and is currently supported for the hg19 assembly only.

`exploratory` = `TRUE` Flag indicating if the exploratory analysis module is to be executed.

`exploratory.columns` = `NULL` Traits, given as column names or indices in the sample annotation table, to be used in the exploratory analysis. These traits are used in multiple steps in the module: they are visualized using point types and colors in the dimension reduction plots; tested for strong correlations and associations with principal components in a methylation space; used to define groups when plotting beta distributions and/or inter-sample methylation variability. The default value of this parameter - `NULL` - indicates that columns should be automatically selected; see `rnb.sample.groups` for how this is done.

`exploratory.top.dimensions` = `0` Number of most variable probes, sites or regions to select prior to performing dimension reduction techniques and tests for associations. Preselection can significantly reduce the running time and memory usage in the exploratory analysis module. Setting this number to zero (default) disables preselection.

`exploratory.principal.components` = `8` Maximum number of principal components to be tested for associations with other factors, such as control probe states and sample traits. This must be an `integer` value between `0` and `10`. Setting this option to `0` disables such tests.

`exploratory.correlation.pvalue.threshold` = `0.01` Significance threshold for a p-value resulting from applying a test for association. This is a value between 0 and 1.

`exploratory.correlation.permutations` = `10000` Number of permutations in tests performed to check for associations between traits, and between control probe intensities and coordinates in the prinicipal component space. This must be a non-negative `integer`. Setting this option to `0` disables permutation tests.

`exploratory.correlation.qc` = `TRUE` [Infinium 450k] Flag indicating if quality-associated batch effects should be studied. This amounts to testing for associations between intensities of quality control probes and principal components. This option has effect only when `exploratory.principal.componer` is non-zero.

`exploratory.beta.distribution` = `TRUE` Flag indicating whether beta value distributions for sample groups and probe or site categories should be computed.

`exploratory.intersample` = `FALSE` Flag indicating if methylation variability in sample groups should be computed as part of the exploratory analysis module. If NULL (default), the plots are created for Bead Array data sets and deactivated for sequencing data sets.

`exploratory.deviation.plots` = `FALSE` Flag indicating if the inter-sample methylation variability step in the exploratory analysis module should include deviation plots. Deviation plots show intra-group methylation variability at the covered sites and regions. Setting this option to `NULL` (default) enables deviation plots on Infinium datasets, but disables them in case of sequencing-based datasets, because their generation can be very computationally intensive. This option has effect only when `exploratory.intersample` is `TRUE`.

`exploratory.clustering` = `"all"` Which sites should be used by clustering algorithms in the exploraroty analysis module. **RnBeads** performs several algorithms that cluster the samples in the dataset. If this option is set to `"all"` (default), clustering is performed using all sites; a value of `"top"` indicates that only the most variable sites are used (see the option `exploratory.clustering.top.sites`); and `"none"` disables clustering.

`exploratory.clustering.top.sites` = `1000` Number of most variable sites to use when visualizing heatmaps. This must be a non-empty `integer` vector containing positive values. This option is ignored when `exploratory.clustering` is `"none"`.

`exploratory.clustering.heatmaps.pdf` = `FALSE` Flag indicating if the generated methylation value heatmaps in the clustering section of the exploratory analysis module should be saved as

PDF files. Enabling this option is not recommended for large values of `exploratory.clustering.top.sites` (more than 200), because heatmaps might generate very large PDF files.

`exploratory.region.profiles` = `""` Region types for generating regional methylation profiles. If `NULL` (default), regional methylation profiles are created only for the region types that are available for the targeted assembly and summarized in the dataset of interest. Setting this option to an empty vector disables the region profiles step in the exploratory analysis module.

`exploratory.gene.symbols` = `NULL` A list of gene symbols to be used for custom locus profiling. Locus views will be generated for these genes.

`exploratory.custom.loci.bed` = `NULL` Path to a bed file containing custom genomic regions. Locus views will be generated for these regions.

`nv.heatmap` = `TRUE` [Microarrays] Flag indicating if a heatmap of the beta values for all nv probes is to be generated.

`nv.beta.distribution` = `TRUE` [Microarrays] Flag indicating if a density plot of the beta values for all nv probes is to be generated.

`differential` = `TRUE` Flag indicating if the differential methylation module is to be executed.

`differential.site.test.method` = `"limma"` Method to be used for calculating p-values on the site level. Currently supported options are "ttest" for a (paired) t-test and "limma" for a linear modeling approach implemented in the `limma` package for differential expression in microarrays.

`differential.variability` = `FALSE` Flag indicating if differential variability analysis is to be conducted. If TRUE, the method specified in `differential.variability.method` is applied to detect sites that show differential variability between the groups that are specified.

`differential.variability.method` = `"diffVar"` Method to be used for calculating p-values on the differential variable sites. Currently supported options are "diffVar" implemented in the `missMethyl` package and "iEVORA".

`differential.permutations` = `0` Number of permutation tests performed to compute the p-value of rank permutation tests in the differential methylation analysis. This must be a non-negative `integer`. Setting this option to `0` (default) disables permutation tests for rank permutations. Note that p-values for differential methylation are computed and also considered for the ranking in any case.

`differential.comparison.columns` = `NULL` Column names or indices in the table of the sample annotation table to be used for group definition in the differential methylation analysis. The default value - `NULL` - indicates that columns should be automatically selected. See `rnb.sample.groups` for how this is done. By default, the comparisons are done in a one vs. all manner if there are multiple groups defined in a column.

`differential.comparison.columns.all.pairwise` = `NULL` Column names or indices in the table of sample annotation table to be used for group definition in the differential methylation analysis in which all pairwise comparisons between groups should be conducted (the default is one vs all if multiple groups are specified in a column). Caution: for large numbers of sample groups this can lead to combinatorial explosion and thus to huge runtimes. A value of `NULL` (default) indicates that no column is selected for all pairwise comparisons explicitely. If specified, the selected columns must be a subset of the columns that will be selected according to the `differential.comparison.columns` option.

`covariate.adjustment.columns` = `NULL` Column names or indices in the table of phenotypic information to be used for confounder adjustment in the differential methylation analysis. Currently this is only supported for `differential.site.test.method=="limma"`.

`columns.pairing` = `NULL` A NAMED vector containing for each column name for which paired analysis should be performed (say columnA) the name or index of another column (say columnB) in which same values indicate the same pairing. columnA should be the name of the value columnB in this vector. For more details see `rnb.sample.groups` `differential.adjustment.sva` = `FALSE` Flag indicating if the differential methylation analysis should account for Surrogate Variables. If `TRUE`, **RnBeads** looks for overlaps between the `differential.comparison.columns` and `inference.targets.sva` options and include the surrogate variables as confounding factors only for these columns. In other words, it will only have an effect if the corresponding inference option (see `inference.targets.sva` option for details) is enabled. Currently this is only supported for `differential.site.test.method=="limma"`.

`differential.adjustment.celltype` = `FALSE` Should the differential methylation analysis account for celltype using the reference based Houseman method. It will only have an effect if the corresponding inference option is enabled (see `inference.reference.methylome.column` option for details). Currently this is only supported for `differential.site.test.method=="limma"`.

`differential.enrichment.go` = `FALSE` Flag indicating whether Gene Ontology (GO)-enrichment analysis is to be conducted on the identified differentially methylated regions.

`differential.enrichment.lola` = `FALSE` Flag indicating whether `LOLA`-enrichment analysis is to be conducted on the identified differentially methylated regions.

`differential.enrichment.lola.dbs` = `c("${LOLACore}")` Vector of directories containing LOLA databases. The following placeholders are allowed which will automatically download corresponding databases from the internet: `"${LOLACore}"` and `"${LOLAExt}"` for the Core and Extended LOLA Databases respectively.

`differential.report.sites` = `TRUE` Flag indicating whether a section corresponding to differential site methylation should be added to the report. Has no effect on the actual analysis, just the report. To disable differential site methylation analysis entirely use the `analyze.sites` option.

`export.to.bed` = `FALSE` Flag indicating whether the data should be exported to bed files.

`export.to.trackhub` = `NULL` `character` vector specifying which data types should be exported to Track hub directories. Possible values in the vector are `"bigBed"` and `"bigWig"`. When this options is set to `NULL`, track hub export is disabled. Note that if `"bigBed"` is contained in this option, bed files are created automatically.

`export.to.csv` = `FALSE` Flag indicating whether methylation value matrices are to be exported to comma-separated value (CSV) files.

`export.to.ewasher` = `FALSE` Flag indicating whether methylation values and differential methylation analysis settings should be exported to a format compatible with FaST-LMM-EWASher, a tool for adjusting for cell-type compositions. See Zou, J., et al., Nature Methods, 2014 for further details on the tool.

`export.types` = `"sites"` `character` vector of sites and region names to be exported. If `NULL`, no region methylation values are exported.

`disk.dump.big.matrices` = `TRUE` Flag indicating whether big tables should be stored on disk rather than in main memory in order to keep memory requirements down. May slow down analysis!

`logging.exit.on.error` = `FALSE` Flag indicating if the active R session should be terminated when an error is encountered during execution.

`distribution.subsample` = `1000000` When plotting methylation value distributions, this threshold specifies the number of observations drawn per group. Distributions are estimated and plotted based on these random subsamples. This approach can significantly reduce the memory requirements of the preprocessing and exploratory analysis modules, where methylation value distributions are plotted. Setting this to `0` disables subsampling. More information is presented the Details section of `rnb.step.betadistribution`.

`enforce.memory.management` = `FALSE` Flag indicating whether in some places of the code memory management should actively being enforced in order to achieve a better memory profile. I.e. garbage collection, variable removal is conducted actively. May slow down analysis.

`enforce.destroy.disk.dumps` = `FALSE` Flag indicating whether disked dumped big matrices (see `disk.dump.big.matrices` option) should actively be deleted when RnBSets are modified. You should switch it to `TRUE` when `disk.dump.big.matrices` is `TRUE` and the amount of hard drive space is also limited.

Author(s)

Yassen Assenov

Examples

```
str(rnb.options())
rnb.getOption("filtering.greedycut")
```

`rnb.options2xml` *rnb.options2xml*

Description

Exports all option values to an XML document.

Usage

```
rnb.options2xml(pretty = TRUE)
```

Arguments

`pretty`	Flag indicating if the document should be formatted to be easily readable. For example, if this is set to `TRUE` (default), every element is located on separate line. Formatting does not affect the validity of the generated XML tree.

Value

XML document in the form of a `character` that encodes all options and their current values.

Author(s)

Yassen Assenov

Examples

```
cat(rnb.options2xml(), file = "rnbeads_options.xml")
```

`rnb.performance.profile`

rnb.performance.profile

Description

Enables one of the pre-installed anlaysis option profiles.

Usage

```
rnb.performance.profile(data.type = "450k", profile)
```

Arguments

`data.type`	Type of dataset targeted; this must be one of `"450k"` (default) or `"bs"`.
`profile`	Option profile; this must be one of `"minimal"`, `"moderate"` or `"full"`.

Value

Invisibly, a `list` containing the previous values of all modified options.

Author(s)

Pavlo Lutsik

`rnb.plot.beta.comparison`

rnb.plot.beta.comparison

Description

Draws plots that compare two distributions of beta values.

Usage

```
rnb.plot.beta.comparison(
  beta.values,
  fprefix,
  report = NULL,
  qq.length = 501L,
  points.per.group = rnb.getOption("distribution.subsample")
)
```

Arguments

`beta.values`	Two beta value sequences in the form of a named `list` of two non-empty vectors of type `double`. If any of the vectors contains `NA`s, this method may exit with an error.
`fprefix`	File name prefix for the plots. This function appends the suffixes `"_density"`, `"_histogram"` and `"_qq"` to this prefix.
`report`	Report to which the plots are to be added.
`qq.length`	Positive `integer` value showing the number of quantiles to be calculated and presented in the generated Q-Q plot.
`points.per.group`	Maximum number of values to use in plotting a group's distribution. Groups that contain more observations than this threshold are subsampled. Setting this parameter to a value less than `2` disables subsampling.

Value

List of all generated plots, each being an object ot type `ReportPlot`.

Author(s)

Yassen Assenov

`rnb.plot.betadistribution.probeCategories` *rnb.plot.betadistribution.probeCategories*

Description

plot beta value distrubions given probe categories

Usage

```
rnb.plot.betadistribution.probeCategories(
  beta.matrix,
  probe.cat,
  annotation = "Group",
  color.legend = NULL,
  log.str = NULL,
  points.per.group = rnb.getOption("distribution.subsample")
)
```

Arguments

`beta.matrix`	Beta values in the form of a non-empty `matrix` of type `double`. Rows in this matrix must correspond to Infinium probes, and columns - to samples.
`probe.cat`	`factor` vector of length `nrow(beta.matrix)` corresponding to the probe categories.
`annotation`	Name of the annotation being visualized, in the form of a `character` vector of length 1.

`color.legend`	Color legend to use in the form of a `character` vector with element names. The values in this vector should encode colors. All values in `probe.cat` must be present in the names of this color legend. If this parameter is `NULL`, a default color legend is be constructed.
`log.str`	string specifying more details for the log file
`points.per.group`	the targeted number of points per group. Set this to a value < 1 to disable subsampling. More information in the Details section of `rnb.step.betadistribution`

Value

The plot as a `ggplot2` object.

Author(s)

Fabian Mueller

See Also

rnb.plot.betadistribution.sampleGroups

Examples

```
library(RnBeads.hg19)
data(small.example.object)
logger.start(fname=NA)
meth.mat <- meth(rnb.set.example)
probe.types <- annotation(rnb.set.example)[, "Design"]
rnb.plot.betadistribution.probeCategories(meth.mat,probe.types,annotation="Infinium probe type")
```

`rnb.plot.betadistribution.sampleGroups`
rnb.plot.betadistribution.sampleGroups

Description

Plots beta value distrubions given a sample grouping.

Usage

```
rnb.plot.betadistribution.sampleGroups(
  beta.matrix,
  sample.group.inds,
  annotation = "Group",
  log.str = NULL,
  points.per.group = rnb.getOption("distribution.subsample")
)
```

Arguments

`beta.matrix`	Beta values in the form of a non-empty `matrix` of type `double`. Rows in this matrix must correspond to Infinium probes, and columns - to samples.
`sample.group.inds`	Named `list` that contains indices for the samples contained in the groups in `beta.matrix`. The number of groups is determined by the length of the list, and its names are used as group names.
`annotation`	Name of the annotation being visualized, in the form of a `character` vector of length 1.
`log.str`	string specifying more details for the log file
`points.per.group`	the targeted number of points per group. Set this to a value < 1 to disable subsampling. More information in the Details section of `rnb.step.betadistribution`

Value

the plot as a `ggplot2` object

Author(s)

Fabian Mueller

See Also

rnb.plot.betadistribution.probeCategories

Examples

```
library(RnBeads.hg19)
data(small.example.object)
logger.start(fname=NA)
meth.mat <- meth(rnb.set.example)
sample.groups <- rnb.sample.groups(rnb.set.example)[[1]]
rnb.plot.betadistribution.sampleGroups(meth.mat,sample.groups)
```

`rnb.plot.biseq.coverage`

rnb.plot.biseq.coverage

Description

Plots the sequencing coverage of the RnBiseqSet object across the genomic coordinate

Usage

```
rnb.plot.biseq.coverage(
  rnbs.set,
  sample,
  type = "sites",
  writeToFile = FALSE,
  numeric.names = FALSE,
  covg.lists = NULL,
  ...
)
```

Arguments

`rnbs.set`	RnBiseqSet object
`sample`	unique sample identifier. In case `rnb.getOption("identifiers.column")` is not `NULL`, `sample` should attain values from the corresponding column, or `colnames(meth(rnb.set))` otherwise
`type`	`character` singleton. If `site` the coverage information is plotted for each methylation site. Otherwise should be one of the regions returned by `rnb.region.types`
`writeToFile`	flag specifying whether the output should be saved as `ReportPlot`
`numeric.names`	if `TRUE` and `writeToFile` is `TRUE`substitute the plot options in the plot file name with digits
`covg.lists`	if available, the output of `rnb.execute.quality`
`...`	other arguments to `createReportPlot`

Value

plot as an object of type `ReportPlot` if `writeToFile` is `TRUE` and of class `ggplot` otherwise.

Author(s)

Pavlo Lutsik

`rnb.plot.biseq.coverage.hist`

rnb.plot.biseq.coverage.hist

Description

Plots the histograms of the coverage

Usage

```
rnb.plot.biseq.coverage.hist(
  rnbs.set,
  sample,
  type = "sites",
  writeToFile = FALSE,
  numeric.names = FALSE,
```

```
  covg.max.percentile = 1,
  ...
)
```

Arguments

`rnbs.set`	RnBiseqSet object
`sample`	unique sample identifier. In case `rnb.getOption("identifiers.column")` is not `NULL`, `sample` should attain values from the corresponding column, or `colnames(meth(rnb.set))` otherwise
`type`	`character` singleton. If `site` the coverage information is plotted for each methylation site. Otherwise should be one of the regions returned by `rnb.region.types`
`writeToFile`	a flag specifying whether the output should be saved as `ReportPlot`
`numeric.names`	if `TRUE` and `writeToFile` is `TRUE`substitute the plot options in the plot file name with digits
`covg.max.percentile`	the maximum percentile of the coverage to be plotted
`...`	other arguments to `createReportPlot`

Value

plot as an object of type `ReportPlot` if `writeToFile` is `TRUE` and of class `ggplot` otherwise.

Author(s)

Pavlo Lutsik

`rnb.plot.biseq.coverage.violin`
rnb.plot.biseq.coverage.violin

Description

Plots the violin plots of the coverage distribution

Usage

```
rnb.plot.biseq.coverage.violin(
  rnbs.set,
  samples,
  fname = NULL,
  type = "sites",
  covg.range = NULL,
  ...
)
```

Arguments

`rnbs.set`	RnBiseqSet object
`samples`	unique sample identifiers. In case `rnb.getOption("identifiers.column")` is not `NULL`, `samples` should attain values from the corresponding column, or `colnames(meth(rnb.set))` otherwise
`fname`	base filename for the files to be plotted. If NULL, the plot will not be written to file
`type`	`character` singleton. If `site` the coverage information is plotted for each methylation site. Otherwise should be one of the regions returned by `rnb.region.types`
`covg.range`	Vector of length 2 specifying the range of coverage to be plotted. if `NULL` (default) the entire range will be plotted
`...`	other arguments to `createReportPlot`

Value

plot as an object of type `ReportPlot` if `writeToFile` is `TRUE` and of class `ggplot` otherwise.

Author(s)

Fabian Mueller

`rnb.plot.control.barplot`

rnb.plot.control.barplot

Description

Per-sample bar plots of Illumina HumanMethylation control probes

Usage

```
rnb.plot.control.barplot(
  rnb.set,
  probe,
  sample.subset = 1:length(samples(rnb.set)),
  writeToFile = FALSE,
  numeric.names = FALSE,
  name.prefix = NULL,
  verbose = FALSE,
  ...
)
```

Arguments

`rnb.set`	`RnBeadRawSet` or `RnBeadSet` object with valid quality control information
`probe`	exact id of the control probe consisting of the control probe type (see `rnb.plot.control.boxplot`)
`sample.subset`	an integer vector specifying the subset of samples for which the plotting should be performed

`writeToFile`	flag specifying whether the output should be saved as `ReportPlot`
`numeric.names`	if `TRUE` and `writeToFile` is `TRUE`substitute the plot options in the plot file name with digits
`name.prefix`	in case `writeToFile` is `TRUE`, a `character` singleton specifying a prefix to the variable part of the image file names
`verbose`	if `TRUE` additional diagnostic output is generated
`...`	other arguments to `createReportPlot`

Value

plot as an object of type `ReportPlot` if `writeToFile` is `TRUE` and of class `ggplot` otherwise.

Author(s)

Pavlo Lutsik

Examples

```
library(RnBeads.hg19)
data(small.example.object)
control.meta.data <- rnb.get.annotation("controls450")
ctrl.probe<-paste0(unique(control.meta.data[["Target"]])[4], ".3")
print(ctrl.probe) # EXTENSION.3
rnb.plot.control.barplot(rnb.set.example, ctrl.probe)
```

`rnb.plot.control.boxplot`

rnb.plot.control.boxplot

Description

Box plots of various control probes

Usage

```
rnb.plot.control.boxplot(
  rnb.set,
  type = rnb.infinium.control.targets(rnb.set@target)[1],
  writeToFile = FALSE,
  numeric.names = FALSE,
  ...
)
```

Arguments

`rnb.set`	`RnBeadRawSet` or `RnBeadSet` object with valid quality control information.
`type`	type of the control probe; must be one of the `"BISULFITE CONVERSION I"`, `"BISULFITE CONVERSION II"`, `"EXTENSION"`, `"HYBRIDIZATION"`, `"NEGATIVE"`, `"NON-POLYMORPHIC"`, `"NORM_A"`, `"NORM_C"`, `"NORM_G"`, `"NORM_T"`, `"SPECIFICITY I"`, `"SPECIFICITY II"`, `"STAINING"`, `"TARGET REMOVAL"`.
`writeToFile`	flag specifying whether the output should be saved as `ReportPlot`
`numeric.names`	if `TRUE` and `writeToFile` is `TRUE`substitute the plot options in the plot file name with digits
`...`	other arguments to `createReportPlot`

Value

plot as an object of type `ReportPlot` if `writeToFile` is `TRUE` and of class `ggplot` otherwise.

Author(s)

Pavlo Lutsik

Examples

```
library(RnBeads.hg19)
data(small.example.object)
rnb.plot.control.boxplot(rnb.set.example)
```

`rnb.plot.coverage.thresholds`
rnb.plot.coverage.thresholds

Description

Plots the number of remaining CpGs after applying different thresholds for coverage and support.

Usage

```
rnb.plot.coverage.thresholds(rnb.set, min.coverages, fname = NA, ...)
```

Arguments

`rnb.set`	Methylation dataset as an object of type `RnBiseqSet`.
`min.coverages`	Non-empty `integer` vector storing the unique positive cutoff values to be applied for minimal coverage. Names, if present, are interpreted as colors that must be used to denote the corresponding values.
`fname`	File name to save the generated plot to. See the *Details* section for restrictions.
`...`	Additional named parameters related to saving the plot to files. These can include: `report`, `width`, `height`, `create.pdf`, `low.png` and `high.png`. These parameters are ignored when `fname` is `NULL` or `NA`.

Details

If `fname` is specified, this function calls `createReportPlot` to save the plot to PDF and/or PNG files. See its documentation for information on acceptable file names. Additional parameters - `report`, `width`, `height`, etc. - can also be given. If image width is not specified, it is set to a value between 4.7 and 9.2 (inches), depending on the number of samples in the dataset. The default image height is fixed to 7.2.

Value

If `fname` is `NULL` or `NA` (default), the generated plot as an object of type `ggplot2`; otherwise, the initialized and closed `ReportPlot` object, invisibly.

Author(s)

Yassen Assenov

`rnb.plot.ct.heatmap` *rnb.plot.ct.heatmap*

Description

Plot contributions of the cell types

Usage

```
rnb.plot.ct.heatmap(ct.obj, type = "nonnegative", writeToFile = FALSE, ...)
```

Arguments

`ct.obj`	Object of class `CellTypeInferenceResult` as returned by rnb.execute.ct.estimation.
`type`	Type of cell type contributions to plot.
`writeToFile`	If `TRUE`, the plot will be written to a file.
`...`	Other arguments passed to `createReportPlot`.

Details

The cell type contributions are visualized as a heatmap

Value

if `writeToFile=TRUE` an object of class `ReportPlot`, or the protted matrix otherwise

Author(s)

Pavlo Lutsik

`rnb.plot.dreduction` *rnb.plot.dreduction*

Description

Creates a dimension reduction plot based on the methylation values of the given dataset.

Usage

```
rnb.plot.dreduction(
  rnb.set,
  plot.type = "pca",
  dimensions = 1:2,
  distance.metric = "euclidean",
  target = "sites",
  point.types = 0L,
  point.colors = 0L,
  legend.space = 2
)
```

Arguments

`rnb.set`	Methylation dataset as an object of type inheriting `RnBSet`. This dataset must contain at least four samples.
`plot.type`	Type of plot to be created. This must be one of `"pca"` (projection to two principal components), `"mds"` (multidimensional scaling to two dimensions) or `"tsne"` (t-distributed stochastic neighbor embedding to two dimensions). The section *Details* provides more details on how the dimension reduction techniques are applied.
`dimensions`	Vector of two positive `integer` values giving the principle components to be shown in the horizontal and vertical axis of the plot. This parameter is considered only when `plot.type` is `"pca"`.
`distance.metric`	Distance metric to be applied when reducing the dimensionality of the methylation data. This must be one of `"eucledian"` or `"manhattan"`. The second metric is not supported by principal component analysis.
`target`	Site or region type to be used in the dimension reduction technique. This must be either `"sites"` (individual CpGs) or one of the region types summarized in `rnb.set`.
`point.types`	Trait, specified as column name or index in the sample annotation table of `rnb.set`, to be used to define point types in the plot. Setting this parameter to zero (default) or to a trait that does not define categories results in all samples being displayed as filled circles. If this parameter specifies a column that can be used as sample identifiers, the plot displays the samples as identifiers instead of points.
`point.colors`	Trait, specified as column name or index in the sample annotation table of `rnb.set`, to be used to define sample colors in the plot. Setting this parameter to zero (default) or to a trait that does not define numerical values or categories results in all samples being displayed in black.

`legend.space`	Width, in inches, of the space dedicated for legends that will be assigned on the right side of the plot. This parameter is considered only if legends are actually included, that is, if sample traits are mapped to point types and/or colors.

Details

The analysis option `"exploratory.top.dimensions"` controls whether dimension reduction is applied on all probes, sites or regions available in the given dataset, or only on the most variable ones. In case a trait is mapped to point types, the shapes to use are taken from the option `"points.category"`. Similary, the option `"colors.category"` determines which colors are used when mapping sample categories to color. In cases when numerical values are mapped to color, the option `"colors.3.gradient"` is used. If the set of value contains both positive and negative numbers, the middle point in the color legend is set to zero. See *RnBeads Options* for more information on the options mentioned above.

Value

The generated plot as an object of type `ggplot`. The object also contains an attribute `"info"`, which is a list with the following elements:

`"Target"` Targeted sites or regions; the value of the parameter `target`.

`"Technique"` Dimension reduction technique applied; one of `"PCA"` or `"MDS"`.

`"All"` Total number of sites or regions defining the high dimensional methylation space.

`"Missing"` Number of dimensions ignored because they contain (only) missing values.

`"Selected"` Number of dimensions used when applying a dimension reduction technique.

`"Explained"` Value between `0` and `1` showing the variance explained by the selected dimensions, as a fraction of the total variance of all dimensions.

Author(s)

Yassen Assenov

See Also

`summarized.regions` for listing all region types summarized in a dataset

Examples

```
library(RnBeads.hg19)
data(small.example.object)
pdf("PCA.pdf", width = 7.2, height = 5.2)
print(rnb.plot.dreduction(rnb.set.example, point.colors="Sample_Group"))
dev.off()
```

`rnb.plot.locus.profile` *rnb.plot.locus.profile*

Description

Computes methylation distributions for various region types and sample groups

Usage

```
rnb.plot.locus.profile(
  rnbSet,
  chrom,
  start,
  end,
  grps = NULL,
  plot.m.regions = NULL,
  plot.m.heatmap = TRUE,
  plot.m.smooth = TRUE,
  cvals.grps = rnb.getOption("colors.category"),
  cvals.meth = rnb.getOption("colors.meth"),
  smooth.profile = "wide"
)
```

Arguments

`rnbSet`	RnBSet object
`chrom`	chromosome of window to plot
`start`	start coordinate of window to plot
`end`	end coordinate of window to plot
`grps`	a list of indices for each group to be compared or NULL if no sample grouping information should be displayed
`plot.m.regions`	character vector of region types whose methylation values should be displayed If `grps` is not `NULL` the methylation values will be separated by sample groups.
`plot.m.heatmap`	flag indicating whether sites methylation values should be displayed in a heatmap. If `grps` is not `NULL` the heatmaps will be separated by sample groups.
`plot.m.smooth`	flag indicating whether a scatterplot with smoothing curves should be displayed. If `grps` is not `NULL` the colors will be used to separate sample groups.
`cvals.grps`	colors to be used for the different groups
`cvals.meth`	colors to be used for methylation values and heatmaps
`smooth.profile`	profile to be used for the smoothing curves. Allowed values include `wide` (default) which yields smoother curvers and `narrow` which yields more "wiggly" curves

Value

a `ggplot2` plot object containing the plot

Author(s)

Fabian Mueller

Examples

```
#see RnBeads vignette (section: 'Generating Locus Profile Plots') for examples
```

`rnb.plot.marker.fstat` *rnb.plot.marker.fstat*

Description

Plot the the cell type marker selection based on the reference methylome data

Usage

```
rnb.plot.marker.fstat(ct.object, writeToFile = FALSE, ...)
```

Arguments

`ct.object`	Object of class `CellTypeInferenceResult` as returned by rnb.execute.ct.estimation.
`writeToFile`	If `TRUE`, the plot will be written to a file.
`...`	Other arguments to `createReportPlot`.

Details

The F-statistic values from the cell type association model (first part of eqn. (1) in [1]) are plotted in decreasing order for all tested CpG positions. A vertical line gives a cut-off for the number of selected cell type markers.

Value

if `writeToFile=TRUE` an object of class `ReportPlot`, and the plotted reordered F-statistics vector otherwise

Author(s)

Pavlo Lutsik

References

1. Houseman, Eugene and Accomando, William and Koestler, Devin and Christensen, Brock and Marsit, Carmen and Nelson, Heather and Wiencke, John and Kelsey, Karl. DNA methylation arrays as surrogate measures of cell mixture distribution. BMC Bioinformatics 2012, 13:86

`rnb.plot.negative.boxplot`

rnb.plot.negative.boxplot

Description

Box plots of negative control probes

Usage

```
rnb.plot.negative.boxplot(
  rnb.set,
  sample.subset = 1:length(samples(rnb.set)),
  writeToFile = FALSE,
  name.prefix = NULL,
  ...
)
```

Arguments

`rnb.set`	`RnBeadSet` object with valid quality control information
`sample.subset`	an integer vector specifying the subset of samples for which the plotting should be performed
`writeToFile`	flag specifying whether the output should be saved as `ReportPlot`
`name.prefix`	in case `writeToFile` is `TRUE`, a `character` singleton specifying a prefix to the variable part of the image file names
`...`	other arguments to `createReportPlot`

Value

plot as an object of type `ReportPlot` if `writeToFile` is `TRUE` and of class `ggplot` otherwise.

Author(s)

Pavlo Lutsik

Examples

```
library(RnBeads.hg19)
data(small.example.object)
rnb.plot.negative.boxplot(rnb.set.example)
```

`rnb.plot.num.sites.covg` *rnb.plot.num.sites.covg*

Description

plot the number of sites vs median and other percentiles of coverage

Usage

```
rnb.plot.num.sites.covg(
  rnbs,
  addSampleNames = (length(samples(rnbs)) < 100),
  bar.percentiles = c(0.25, 0.75)
)
```

Arguments

`rnbs`	RnBiseqSet object
`addSampleNames`	should the sample names be added to the plot
`bar.percentiles`	the percentiles to be used for the error bars. Must be a vector of length 2 of which the first two elements will be used

Value

plot as an object of type `ggplot`

Author(s)

Fabian Mueller

`rnb.plot.nv.heatmap` *rnb.plot.nv.heatmap*

Description

Heatmap of beta values from nv probes.

Usage

```
rnb.plot.nv.heatmap(dataset, writeToFile = FALSE, ...)
```

Arguments

`dataset`	Dataset as an object of type inheriting `RnBeadSet`, or a matrix of methylation beta values.
`writeToFile`	Flag specifying whether the output should be saved as `ReportPlot`.
`...`	Additional named arguments passed to `createReportPlot`. These are used only if `writeToFile` is `TRUE`.

Value

If `writeToFile` is `TRUE`, plot as an object of type `ReportPlot`. Otherwise, there is no value returned (invisible `NULL`).

Author(s)

Pavlo Lutsik, Baris Kalem

`rnb.plot.pheno.categories` *rnb.plot.pheno.categories*

Description

Generates bar charts summarizing the categorical traits in a sample annotation table.

Usage

```
rnb.plot.pheno.categories(
  annotations,
  columns = NULL,
  fileprefix = "barchart_pheno",
  report = NULL,
  color.values = rnb.getOption("colors.category")
)
```

Arguments

`annotations`	Methylation dataset as an object of type inheriting `RnBSet`, or its sample annotations in the form of a `data.frame`. If this parameter is a dataset, the annotation information is extracted using the method `pheno`.
`columns`	Optional; predefined column names (in the form of a `character` vector) or indices (an `integer` vector) to consider. All other columns in the annotation table will be ignored.
`fileprefix`	`character` vector with one element storing the file name prefix of the output files, without the extension. Only a limited set of symbols is allowed to be used in this prefix.
`report`	Report to contain the generated plots. If specified, this must be an object of type `Report`.
`color.values`	Non-empty `character` vector containing the color scheme to be mapped to the categories defined in the annotation table. Colors are recycled if necessary, that is, if the length of this vector is smaller than the number of categories in a trait.

Details

This function identifies the traits that define sample subgroups and then generates one report plot per trait. Every report plot consists of two files. File names are formed by appending an index and file extension to `fileprefix`. Thus, the suffixes appended are `"_1.pdf"`, `"_1.png"`, `"_2.pdf"`, `"_2.png"`, ... Existing files with the generated filenames are overwritten.

Value

List of report plots. The names in this list are the column names in the annotation table that were selected for visualization. In case no suitable categorical traits are found among the provided annotations, this function returns an empty list.

Author(s)

Yassen Assenov

See Also

`rnb.sample.groups` for identifying traits in the annotation table that define sample subgroups; `createReportPlot` for the allowed symbols to be used in `fileprefix`

`rnb.plot.region.profile.density` *rnb.plot.region.profiles*

Description

Plots the density of methylation levels accross all regions of the specified type

Usage

```
rnb.plot.region.profile.density(
  rnb.set,
  sample,
  region.type = "",
  region.profile = NULL,
  extend.by = 0.33
)
```

Arguments

`rnb.set`	RnBSet object
`sample`	Index or name of the sample for which the plot should be generated
`region.type`	Region type for which the plot should be generated
`region.profile`	Alternative to specifying `region.type`, the function can accept a region profile generated by the `rnb.find.relative.site.coord` function
`extend.by`	A number between 0 and 1 specifying the percentage by which a region is extended in order to capture methylation information before region start and after region end

Value

a ggplot2 object for plotting the plot shows the density of methylation levels of sites accross the specified region type for all regions of that type from 0 (region start) to 1 (region end). Sites in the flanking areas are also shown (coordinates <0 and >1).

Author(s)

Fabian Mueller

Examples

```
library(RnBeads.hg19)
data(small.example.object)
logger.start(fname=NA)
rnb.plot.region.profile.density(rnb.set.example,1,"genes")
```

`rnb.plot.region.profiles`

rnb.plot.region.profiles

Description

Creates a composite plot showing the sample and groupwise smoothed estimates of methylation values accross all regions of the specified type

Usage

```
rnb.plot.region.profiles(
  rnb.set,
  group.index.list,
  region.type = "",
  region.profile = NULL,
  extend.by = 0.33,
  cvalues = rnb.getOption("colors.category")
)
```

Arguments

`rnb.set`	RnBSet object
`group.index.list`	a list (preferably named) containing sample indices for each group a list of such lists is for instance generated by the `rnb.sample.groups` function.
`region.type`	Region type for which the plot should be generated
`region.profile`	Alternative to specifying `region.type`, the function can accept a region profile generated by the `rnb.find.relative.site.coord` function
`extend.by`	A number between 0 and 1 specifying the percentage by which a region is extended in order to capture methylation information before region start and after region end
`cvalues`	Color values that will be assigned to sample groups

Value

a ggplot2 object for plotting the plot shows the smoothed methylation levels of sites accross the specified region type for all regions of that type from 0 (region start) to 1 (region end). Sites in the flanking areas are also shown (coordinates <0 and >1). Smoothing is stratified by sample (dashed lines) and sample group (thick solid lines). Cubic splines are used for smoothing

Author(s)

Fabian Mueller

Examples

```
#Careful: this might take a while
library(RnBeads.hg19)
data(small.example.object)
logger.start(fname=NA)
rnb.plot.region.profiles(rnb.set.example,rnb.sample.groups(rnb.set.example)[[1]],"genes")
```

`rnb.plot.region.site.density`

rnb.plot.region.site.density

Description

Plots the density of sites accross the specified region type

Usage

```
rnb.plot.region.site.density(rnb.set, region.type, extend.by = 0.33)
```

Arguments

`rnb.set`	RnBSet object
`region.type`	Region type for which the plot should be generated
`extend.by`	A number between 0 and 1 specifying the percentage by which a region is extended in order to capture methylation information before region start and after region end

Value

a ggplot2 object for plotting the plot shows the density of sites accross the specified region type for all regions of that type from 0 (region start) to 1 (region end). Sites in the flanking areas are also shown (coordinates <0 and >1).

Author(s)

Fabian Mueller

Examples

```
library(RnBeads.hg19)
data(small.example.object)
logger.start(fname=NA)
rnb.plot.region.site.density(rnb.set.example,"genes")
```

`rnb.plot.sentrix.distribution` *rnb.plot.sentrix.distribution*

Description

Creates a point-and-whisker plots showing beta value distributions at Sentrix positions for the given slide.

Usage

```
rnb.plot.sentrix.distribution(rnb.set, sentrix.id)
```

Arguments

`rnb.set`	HumanMethylation450K dataset as an object of type `RnBeadSet`.
`sentrix.id`	Slide number (Sentrix ID) as an `integer` or `character` singleton.

Value

Generated point-and-whisker plot (an instance of `ggplot`) of mean methylations for the samples on the specified slide, or `FALSE` if the dataset is non-empty but does not contain samples on the given slide. If the provided dataset does not contain valid Sentrix ID and position information (or is an empty dataset), this method returns `NULL`.

Author(s)

Yassen Assenov

Examples

```
library(RnBeads.hg19)
data(small.example.object)
sid<-as.character(pheno(rnb.set.example)[["Sentrix_ID"]][1])
rnb.plot.sentrix.distribution(rnb.set.example,sid)
```

`rnb.plot.sentrix.distributions` *rnb.plot.sentrix.distributions*

Description

Creates one or more point-and-whisker plots showing beta value distributions at Sentrix positions.

Usage

```
rnb.plot.sentrix.distributions(rnb.set, fprefix = "sentrix_whisker", ...)
```

Arguments

`rnb.set`	HumanMethylation450K dataset as an object of type `RnBeadSet`.
`fprefix`	File name prefix to be used in the generated plots. In order to ensure independence of the operating system, there are strong restrictions on the name of the file. See the documentation of `createReportPlot` for more information.
`...`	Other arguments passed to `createReportPlot`. These can include the named parameters `report`, `width`, `height`, and others.

Details

If no additional parameters are specified, this function creates one PDF and one low-resolution PNG file for every generated plot.

Value

Point-and-whisker plot (an instance of `ReportPlot`), or a list of such plots - one per slide. If the provided dataset does not contain valid Sentrix ID and position information (or is an empty dataset), this method returns `NULL`.

Author(s)

Yassen Assenov

See Also

`rnb.plot.sentrix.distribution` for creating a single plot for a specified slide number

`rnb.plot.snp.barplot` *rnb.plot.snp.barplot*

Description

Bar plots of beta-values from the genotyping probes

Usage

```
rnb.plot.snp.barplot(
  dataset,
  probeID,
  writeToFile = FALSE,
  numeric.names = FALSE,
  ...
)
```

Arguments

`dataset`	Dataset as an instance of `RnBeadRawSet` or `RnBeadSet`. Alternatively, the dataset can be specified as a non-empty `matrix` containing the computed beta values on the SNP probes.
`probeID`	Probe identifier. This must be one of `rownames(meth(dataset))`.
`writeToFile`	Flag specifying whether the output should be saved as `ReportPlot`.
`numeric.names`	if `TRUE` and `writeToFile` is `TRUE`substitute the plot options in the plot file name with digits.
`...`	Additional named arguments passed to `createReportPlot`.

Value

plot as an object of type `ReportPlot` if `writeToFile` is `TRUE` and of class `ggplot` otherwise.

Author(s)

Pavlo Lutsik

Examples

```
library(RnBeads.hg19)
data(small.example.object)
samp<-samples(rnb.set.example)[1]
rnb.plot.snp.barplot(rnb.set.example, samp)
```

`rnb.plot.snp.boxplot` *rnb.plot.snp.boxplot*

Description

Box plots of beta-values from the genotyping probes

Usage

```
rnb.plot.snp.boxplot(dataset, writeToFile = FALSE, ...)
```

Arguments

`dataset`	Dataset as an object of type inheriting `RnBeadSet`, or a matrix of methylation beta values.
`writeToFile`	Flag specifying whether the output should be saved as `ReportPlot`.
`...`	Additional named arguments passed to `createReportPlot`.

Value

If `writeToFile` is `TRUE`: plot as an object of type `ReportPlot`. Otherwise: plot as an object of type `ggplot`.

Author(s)

Pavlo Lutsik

Examples

```
library(RnBeads.hg19)
data(small.example.object)
rnb.plot.snp.boxplot(rnb.set.example)
```

`rnb.plot.snp.heatmap` *rnb.plot.snp.heatmap*

Description

Heatmap of beta values from genotyping probes.

Usage

```
rnb.plot.snp.heatmap(dataset, writeToFile = FALSE, ...)
```

Arguments

`dataset`	Dataset as an object of type inheriting `RnBeadSet`, or a matrix of methylation beta values.
`writeToFile`	Flag specifying whether the output should be saved as `ReportPlot`.
`...`	Additional named arguments passed to `createReportPlot`. These are used only if `writeToFile` is `TRUE`.

Value

If `writeToFile` is `TRUE`, plot as an object of type `ReportPlot`. Otherwise, there is no value returned (invisible `NULL`).

Author(s)

Pavlo Lutsik

Examples

```
library(RnBeads.hg19)
data(small.example.object)
rnb.plot.snp.heatmap(rnb.set.example)
```

`rnb.read.geo` *Import methylation data from GEO*

Description

Imports Infinium 450K, MethylationEPIC or MethylationEPIC2 data series from the Gene Expression Omnibus. This function uses the series matrix file.

Usage

```
rnb.read.geo(
  accession = NULL,
  verbose = logger.isinitialized(),
  destdir = tempdir()
)
```

Arguments

`accession`	Character string, starting with `"GSE"`, representing the GEO series for download and parsing. Alternatively, this parameter can specify the file name of a previously downloaded GEO series matrix file or its gzipped representation (in which case the filename must end in `".gz"`). Other file formats, such as SOFT files, are not supported.
`verbose`	Flag indicating if messages should be created informing about the progress. If the logger is initialized prior to calling this function, the informative messages are sent to the logger. Warnings and errors are not affected by this parameters, the function always outputs them.
`destdir`	The destination directory for any downloads. Defaults to the (architecture-dependent) temporary directory. Keep in mind that GEO series can be demanding in terms of storage space.

Value

`RnBeadSet` object with phenotypic and beta value information.

Author(s)

Yassen Assenov, modified by Baris Kalem

`rnb.region.types` *rnb.region.types*

Description

Gets the supported region annotations for a given genome assembly.

Usage

```
rnb.region.types(assembly = "hg19")
```

Arguments

`assembly`	Genome assembly of interest. See `rnb.get.assemblies` for the list of supported genomes.

Value

Region types supported by **RnBeads** in the form of a `character` vector. The built-in ones are `"cpgislands"`, `"genes"`, `"promoters"` and `"tiling"`. The names of all custom region definitions are also included in the returned vector.

Author(s)

Yassen Assenov

See Also

`rnb.get.annotation`, `rnb.set.annotation`

Examples

```
"promoters" %in% rnb.region.types() # TRUE
```

`rnb.region.types.for.analysis`

rnb.region.types.for.analysis

Description

Identifies the region types that are summarized by the given dataset and pointed to for analysis.

Usage

```
rnb.region.types.for.analysis(rnb.set)
```

Arguments

`rnb.set`	Methylation dataset as an object of type inheriting `RnBSet`.

Details

This function intersects the value of the analysis option `"region.types"` with the region types that are summarized in the provided dataset. In case the option's value is `NULL`, this function returns all summarized region types in `rnb.set`.

Value

List of all region types to be analyzed in the current dataset in the form of a `character` vector.

Author(s)

Yassen Assenov

See Also

`rnb.getOption` for checking the value of the `"region.types"` option; `summarized.regions` for obtaining the region types summarized in a dataset

Examples

```
library(RnBeads.hg19)
data(small.example.object)
"promoters" %in% rnb.region.types.for.analysis(rnb.set.example)
```

`rnb.remove.annotation` *rnb.remove.annotation*

Description

Deletes a region annotation table. Use this function with caution; its operation cannot be undone.

Usage

```
rnb.remove.annotation(type, assembly = "hg19")
```

Arguments

`type`	One-element `character` vector giving the name of the region annotation.
`assembly`	Genome assembly of interest. See `rnb.get.assemblies` for the list of supported genomes.

Value

Invisibly, `TRUE` if the annotation has been successfully deleted, or `FALSE` if the specified region type is not supported.

Author(s)

Fabian Mueller

See Also

`rnb.get.annotation`, `rnb.region.types`

Examples

```
t.regions <- rnb.get.annotation("tiling")
rnb.remove.annotation("tiling")
```

`rnb.RnBSet.to.bed` *Export to BED files*

Description

Exports the beta values from a methylation dataset to BED files.

Usage

```
rnb.RnBSet.to.bed(
  rnb.set,
  out.dir,
  reg.type = "sites",
  names.quant.meth = TRUE,
  add.track.line = TRUE,
  lexicographic = FALSE,
  verbose = TRUE
)
```

Arguments

`rnb.set`	Methylation dataset as an object of type inheriting `RnBSet`.
`out.dir`	Output directory. If not existing, it will be created. otherwise files in that directory are overwritten.
`reg.type`	Region type to be extracted.
`names.quant.meth`	should the names of the bed regions contain information on the methylation level. If TRUE the following format is applied: meth_percent `covg(rnb.set)` is not NULL
`add.track.line`	Add a track line to the bed file to enable browsers like IGV to display the data better
`lexicographic`	Should lexicographic ordering be used for chromosome names
`verbose`	More detailed logger output

Details

Details on the BED file format can be found in the UCSC Genome Browser documentation. Each methylation site is an entry in the resulting bed file. The Score column corresponds to a site's methylation value in the interval `[0,1]`.

Value

(invisibly) a summary list containing information on the conversion step. elements are `filenames` (a table containing information on which sample has been written to what filename) and `assembly` (a string indicating the assembly used by `rnb.set`).

Author(s)

Fabian Mueller

Examples

```
library(RnBeads.hg19)
data(small.example.object)
logger.start(fname=NA)
rnb.RnBSet.to.bed(rnb.set.example,tempdir())
```

`rnb.RnBSet.to.bedGraph`

rnb.RnBSet.to.bedGraph

Description

Exports the methylation data of an `RnBSet` object to `*.bedGraph` files.

Usage

```
rnb.RnBSet.to.bedGraph(
  rnb.set,
  out.dir = ".",
  reg.type = "sites",
  parameters = character(),
  digits = NULL
)
```

Arguments

`rnb.set`	Dataset as an instance of class `RnBSet`.
`out.dir`	One-element `character` vector signifying the output directory in which to create `bedGraph` files. Setting this to `"."` (default) uses the current working directory. If the output directory does not exist, this function attempts to create it. Any existing files in this directory could be overwritten.
`reg.type`	Site or region type to be exported.
`parameters`	Named `character` vector storing parameters (other than `"type"` and `"name"`) to include in the track definition line. The names of this vector must be the parameter names, and its elements - the corresponding values; missing values (`NA`s) are allowed neither for names, nor for values. This function does not test if all provided parameter names and values conform to the BedGraph track specification.
`digits`	Optionally, number of significant digits after the decimal point to round methylation values to. If specified, this parameter must be an `integer` between `0` and `10`.

Details

The description of the BedGraph track format can be found here. Each methylation site is an entry in the resulting bedGraph file. The Score column corresponds to a site's methylation value in the interval `[0,1]`.

Value

(invisibly) a summary list containing information on the conversion step. elements are `filenames` (a table containing information on which sample has been written to what filename) and `assembly` (a string indicating the assembly used by `rnb.set`).

Author(s)

Fabian Mueller

Examples

```
library(RnBeads.hg19)
data(small.example.object)
logger.start(fname=NA)
rnb.RnBSet.to.bedGraph(rnb.set.example,tempdir())
```

`rnb.RnBSet.to.GRangesList` *rnb.RnBSet.to.GRangesList*

Description

convert an `RnBSet` object to a `GRangesList` object

Usage

```
rnb.RnBSet.to.GRangesList(
  rnb.set,
  reg.type = "sites",
  return.regular.list = FALSE
)
```

Arguments

`rnb.set`	Object of class `RnBSet`
`reg.type`	region type to be converted
`return.regular.list`	flag indicating whether a regular `list` object should be returned instead of a `GRangesList`. Might improve performance in some cases

Value

a `GRangesList` or `list` object with one list element (`GRanges`) for each sample in `rnb.set`

Author(s)

Fabian Mueller

Examples

```
library(RnBeads.hg19)
data(small.example.object)
logger.start(fname=NA)
result <- rnb.RnBSet.to.GRangesList(rnb.set.example)
```

`rnb.run.analysis` *RnBeads Analysis Pipeline*

Description

Starts the **RnBeads** analysis pipeline on the given dataset. It loads the dataset if it is specified as a location.

Usage

```
rnb.run.analysis(
  dir.reports,
  data.source = NULL,
  sample.sheet = NULL,
  data.dir = NULL,
  GS.report = NULL,
  GEO.acc = NULL,
  data.type = rnb.getOption("import.default.data.type"),
  initialize.reports = TRUE,
  build.index = TRUE,
  save.rdata = TRUE
)
```

Arguments

`dir.reports`	Directory to host the generated report files. This must be a `character` of length one that specifies either a non-existent path (when `initialize.reports` is `TRUE`), or an existing directory (when `initialize.reports` is `FALSE`). In the latter case, a call to `rnb.initialize.reports` might be required before viewing the reports.
`data.source`	Methylation dataset as an object of type inheriting `RnBSet`, or a `character` vector specifying the location of the data items on disk. The expected length of the vector differs for different values of `data.type`; see `rnb.execute.import` for a more detailed description. If set, the parameters `sample.sheet`, `data.dir`, `GS.report`, `GEO.acc` will be ignored.
`sample.sheet`	A spreadsheet-like text file with sample annotations. The required columns are different for different values of `data.type`.
`data.dir`	For `data.type %in% c("data.dir", "idat.dir", "bed.dir")` a character singleton specifying the location of the directory with data files. The directory should have zero depth, i.e. should contain no subdirectories.
`GS.report`	GenomeStudio report file. `data.type` will be automatically set to `"GS.report"`.
`GEO.acc`	Gene Expression Omnibus accession of the data series with HumanMethylation450 data. `data.type` will be automatically set to `"GEO"`.

`data.type`	`character` vector of length one specifying the type of the input data. The value must be one of `"data.dir"`, `"idat.dir"`, `"GS.report"`, `"GEO"` or `"rnb.set"`. See `rnb.execute.import` for a more detailed description.
`initialize.reports`	Flag indicating if the report's directory must be initialized. If this parameter is set to `TRUE`, this function attempts to create the path specified by `dir.reports`. Otherwise, `dir.reports` is expected to signify an existing directory.
`build.index`	Flag indicating if a report index file (named `"index.html"`) should be created after all modules in the pipeline complete their analyses. If this is `TRUE`, the index file is also displayed using the function `rnb.show.report`.
`save.rdata`	Flag indicating whether important data objects (the filtered and unfiltered RnBSets, differential methylation) should be saved to an RData file in the reports folder.

Value

Invisibly, the loaded, normalized and/or possibly filtered dataset as an object of type inheriting `RnBSet`.

Author(s)

Yassen Assenov

See Also

RnBeads modules

`rnb.run.dj` *rnb.run.dj*

Description

Starts the RnBeads Data Juggler (RnBeadsDJ) for configuring and running RnBeads analyses from the web browser

Usage

```
rnb.run.dj()
```

Details

A Shiny app is launched in the web browser

Value

Nothing of particular interest

Author(s)

Fabian Mueller

See Also

`rnb.run.analysis` for starting an analysis pipeline

`rnb.run.example` *rnb.run.example*

Description

Executes the analysis pipeline for an example from the RnBeads web site.

Usage

```
rnb.run.example(index = 4L, dir.output = "example")
```

Arguments

`index`	Example to start. This must be one of `1`, `2`, `3` or `4`.
`dir.output`	One-element `character` vector specifying the directory to contain the downloaded data files and generated reports. This must be a non-existent path, as this function attempts to create it.

Details

For more information about the examples, please visit the dedicated page on the RnBeads web site.

Value

Invisibly, the loaded, normalized and/or possibly filtered dataset as an object of type inheriting `RnBSet`.

Author(s)

Yassen Assenov

See Also

`rnb.run.analysis` for starting the analysis pipeline from a local data source

Examples

```
rnb.run.example()
```

`rnb.run.import` *RnBeads Modules in the Analysis Pipeline*

Description

Functions that start the predefined modules in the **RnBeads** analysis pipeline.

Usage

```
rnb.run.import(
  data.source,
  data.type = rnb.getOption("import.default.data.type"),
  dir.reports,
  init.configuration = !file.exists(file.path(dir.reports, "configuration")),
  close.report = TRUE,
  show.report = FALSE
)

rnb.run.qc(
  rnb.set,
  dir.reports,
  init.configuration = !file.exists(file.path(dir.reports, "configuration")),
  close.report = TRUE,
  show.report = FALSE
)

rnb.run.preprocessing(
  rnb.set,
  dir.reports,
  init.configuration = !file.exists(file.path(dir.reports, "configuration")),
  close.report = TRUE,
  show.report = FALSE
)

rnb.run.inference(
  rnb.set,
  dir.reports,
  init.configuration = !file.exists(file.path(dir.reports, "configuration")),
  close.report = TRUE,
  show.report = FALSE
)

rnb.run.tnt(
  rnb.set,
  dir.reports,
  init.configuration = !file.exists(file.path(dir.reports, "configuration")),
  close.report = TRUE,
  show.report = FALSE
)

rnb.run.exploratory(
```

```
  rnb.set,
  dir.reports,
  init.configuration = !file.exists(file.path(dir.reports, "configuration")),
  close.report = TRUE,
  show.report = FALSE
)

rnb.run.differential(
  rnb.set,
  dir.reports,
  init.configuration = !file.exists(file.path(dir.reports, "configuration")),
  close.report = TRUE,
  show.report = FALSE
)
```

Arguments

`data.source`	`character` vector specifying the location of the data items on disk. The expected length of the vector differs for different values of `data.type`; see `rnb.execute.import` for a more detailed description.
`data.type`	`character` vector of length one specifying the type of the input data. The value of this parameter must be one of `"idat.dir"`, `"data.dir"`, `"data.files"`, `"GS.report"`, `"GEO"` or `"rnb.set"`. See `rnb.execute.import` for a more detailed description.
`dir.reports`	Directory to host the generated report file. Note that if this directory contains files, they may be overwritten.
`init.configuration`	Flag indicating if the configuration directory (usually shared among reports) should also be created.
`close.report`	Flag indicating if the created report is to be closed using the `off` method.
`show.report`	Flag indicating if the report is to be displayed after it is created. If this is, `TRUE` `rnb.show.report` is called to open the generated HTML file.
`rnb.set`	Methylation dataset as an object of type inheriting `RnBSet`.

Details

The functions start the import, quality control, preprocessing, covariate inference, tracks and tables, exploratory analysis and differential methylation modules, respectively.

Value

For `rnb.run.import`, `rnb.run.preprocessing` and `rnb.run.inference`, the returned value is a list of two elements - the initialized or modified dataset and the created report. All other functions return the created report, invisibly.

Author(s)

Yassen Assenov

See Also

`rnb.run.analysis` which executes these modules in the order given above

Examples

```
### Running the modules step by step

# Directory where your data is located
data.dir <- "~/RnBeads/data/Ziller2011_PLoSGen_450K"
idat.dir <- file.path(data.dir, "idat")
sample.annotation <- file.path(data.dir, "sample_annotation.csv")

# Directory where the output should be written to
analysis.dir <- "~/RnBeads/analysis"
# Directory where the report files should be written to
report.dir <- file.path(analysis.dir, "reports_details")
rnb.initialize.reports(report.dir)
# Set some analysis options
rnb.options(filtering.sex.chromosomes.removal = TRUE, identifiers.column = "Sample_ID")
## Restrict logging to the console only
logger.start(fname = NA)

## Data import
data.source <- c(idat.dir, sample.annotation)
result <- rnb.run.import(data.source=data.source, data.type="idat.dir", dir.reports=report.dir)
rnb.set <- result$rnb.set

## Quality Control
rnb.run.qc(rnb.set, report.dir)

## Preprocessing
rnb.set <- rnb.run.preprocessing(rnb.set, dir.reports=report.dir)$rnb.set

## Data export
rnb.options(export.to.csv = TRUE)
rnb.run.tnt(rnb.set, report.dir)

## Exploratory analysis
rnb.run.exploratory(rnb.set, report.dir)

## Differential methylation
rnb.run.differential(rnb.set, report.dir)
```

`rnb.run.xml` *rnb.run.xml*

Description

Starts the analysis pipeline from an XML configuration file. This function uses the **XML** package to parse the configuration file.

Usage

```
rnb.run.xml(fname, create.r.command = FALSE)
```

Arguments

`fname`	XML configuration file to read.
`create.r.command`	Flag indicating if the R command(s) that correspond to the given XML configuration should be generated. If this is set to `TRUE`, a file named `"analysis.R"` is created in the reports directory.

Details

Two values are required to be specified (as tags) in the configuration file - `data.source` and `dir.reports`. They define the input and output directory, respectively. In addition, the file may define analysis option values. The vignette *Comprehensive DNA Methylation Analysis with RnBeads* describes in details the syntax of the XML configuration file.

The sample annotation table must be stored as a file in `data.source`. For more information about the required parameters, see the documentation of `rnb.run.analysis`, which is called by this function.

Value

Invisibly, the loaded, normalized and/or possibly filtered dataset as an object of type inheriting `RnBSet`.

Author(s)

Yassen Assenov

See Also

`rnb.run.analysis` for starting an analysis pipeline

`rnb.sample.groups` *rnb.sample.groups*

Description

Identifies sample subgroups defined in the given annotation information.

Usage

```
rnb.sample.groups(
  annotations,
  columns = NULL,
  columns.pairs = NULL,
  min.group.size = rnb.getOption("min.group.size"),
  max.group.count = rnb.getOption("max.group.count")
)
```

Arguments

`annotations`	Methylation dataset as an object of type inheriting `RnBSet`, or its sample annotations in the form of a `data.frame`. If this parameter is a dataset, the annotation information is extracted using the method `pheno`.
`columns`	Optional; predefined column names (in the form of a `character` vector) or indices (an `integer` vector) to consider. All other columns in the annotation table will be ignored.
`columns.pairs`	Optional; a NAMED vector containing for each column name for which paired comparisons should be performed (say columnA) the name or index of another column (say columnB) in which same values indicate the same pairing. columnA should be the name of the value columnB in this vector.
`min.group.size`	Minimum number of samples in each subgroup. This must be a positive `integer`.
`max.group.count`	Maxumum number of subgroups defined by a trait. This must be an `integer` greater than 1.

Value

List of traits that define subgroups in the dataset. For each trait, the defined subgroups are represented by a list of `integer` vectors storing the corresponding sample indices.

Author(s)

Yassen Assenov

Examples

```
library(RnBeads.hg19)
data(small.example.object)
str(rnb.sample.groups(rnb.set.example))
```

`rnb.sample.replicates` *rnb.sample.replicates*

Description

Identifies sample replicates defined in the given sample annotation table.

Usage

```
rnb.sample.replicates(rnb.set, replicate.id.col)
```

Arguments

`rnb.set`	Methylation dataset as an object of type inheriting `RnBSet`.
`replicate.id.col`	Trait (column name in the sample annotation table) that indicates sample replicates. Replicates should have the same value for this trait, while samples without replicates are expected to have unique values or missing values.

Value

List of length of the number of replicates in the dataset. Each element is an `integer` vector storing the corresponding sample indices.

Author(s)

Fabian Mueller

`rnb.sample.summary.table` *rnb.sample.summary.table*

Description

Creates a sample summary table from an RnBSet object

Usage

```
rnb.sample.summary.table(rnbSet)
```

Arguments

`rnbSet`	`RnBSet` of interest.

Value

a summary table (as data.frame) with the following variables for each sample (rows):

`sampleName`	Name of the sample
`*_num (* can be 'sites' or a region type)`	Number of sites or regions with coverage in the sample
`*_covgMean (RnBiseqSet only)`	Mean coverage of sites or regions in the sample
`*_covgMedian (RnBiseqSet only)`	Median coverage of sites or regions in the sample
`*_covgPerc25 (RnBiseqSet only)`	25 percentile of coverage of sites or regions in the sample
`*_covgPerc75 (RnBiseqSet only)`	75 percentile of coverage of sites or regions in the sample
`*_numCovg5, 10, 30, 60 (RnBiseqSet only)`	Number of sites or regions with coverage greater or equal to 5,10,30,60
`sites_numDPval5em2, 1em2, 1em3 (RnBeadSet only)`	Number of sites with a detection p-value smaller than 0.05,0.01,0.001
`**_numSitesMean (** is any region type)`	Mean number of sites in a region
`**_numSitesMedian`	Median number of sites in a region
`**_numSites2, 5, 10, 20`	Number of regions with at least 2,5,10,20 sites with valid methylation measurements

Author(s)

Fabian Mueller

Examples

```
library(RnBeads.hg19)
data(small.example.object)
logger.start(fname=NA)
rnb.sample.summary.table(rnb.set.example)
```

`rnb.save.annotation` *rnb.save.annotation*

Description

Saves the specified region annotation table and its accompanying data structures to a binary file.

Usage

```
rnb.save.annotation(fname, type, assembly = "hg19")
```

Arguments

`fname`	One-element `character` vector giving the name of the file to contain the annotation data. If this file already exists, it will be overwritten.
`type`	One-element `character` vector giving the name of the region annotation.
`assembly`	Genome assembly of interest. See `rnb.get.assemblies` for the list of supported genomes.

Details

This function is used in combination with `rnb.load.annotation` to enable fast reloading of custom region annotations. If can also be used to save a build-in region annotation (e.g. before overwriting it) but not site or control probe annotations.

Value

`TRUE`, invisibly.

Author(s)

Yassen Assenov

See Also

`rnb.load.annotation` for loading a saved annotation

`rnb.section.diffVar.region` *rnb.section.diffVar.region*

Description

Adds information for differentially variable regions to the report.

Usage

```
rnb.section.diffVar.region(
  rnb.set,
  diff.meth,
  report,
  gzTable = FALSE,
  level = 1
)
```

Arguments

`rnb.set`	Object of type `RnBSet` containing methylation information
`diff.meth`	RnBDiffMeth object. See `RnBDiffMeth-class` for details.
`report`	Report object to which the content is added
`gzTable`	Flag indicating if tables should be gzipped
`level`	Which level of section should be created. See `rnb.add.section`.

Value

The modified report object

Author(s)

Michael Scherer

`rnb.set.annotation` *rnb.set.annotation*

Description

Adds or replaces a region annotation table.

Usage

```
rnb.set.annotation(type, regions, description = NULL, assembly = "hg19")
```

Arguments

`type`	One-element `character` vector giving the name of the annotation. If this region type is already available, it will be overwritten for the current session. The type cannot be one of `"CpG"`, `"probes450"` or `"controls450"`, because these names are reserved for the annotation tables of CpG dinucleotides, and Infinium methylation and control probes, respectively.
`regions`	BED file defining regions (see *Details*). Alternatively, the value of this parameter can be a table of genomic regions in the form of a `data.frame`, containing at least the following three columns - `"Chromosome"`, `"Start"` and `"End"` (notice the upper case). The `"chromosome"` column must be a `character` or `factor` vector that lists chromosome names. The `"start"` and `"end"` columns are expected to contain genomic positions as `integers`. The row names of this `data.frame` are used as region identifiers.
`description`	Optional; short description in the form of a non-empty `character` vector. The elements in this vector are concatenated without a separator to form the description of the annotation.
`assembly`	Genome assembly of interest. See `rnb.get.assemblies` for the list of supported genomes.

Details

In case the parameter `regions` specifies an existing BED file, regions are loaded from this file. The number of columns defined must be at least 3. Columns after the sixth one, if present, are dropped. The columns are given the following names: `"chromosome"`, `"start"`, `"end"`, `"id"`, `"score"` and `"strand"`.

The annotation tables in **RnBeads** focus on chromosomes `"chr1"`, `"chr2"`, ..., `"chr22"`, `"chrX"` and `"chrY"`. Regions on other chromosomes are ignored. This function also recognizes the convention of chromosome names such as `"1"`, adopted, for example, by Ensembl. Apart from this, the region definition table is not examined in details by this function; therefore, regions located on unsupported chromosomes or having invalid (e.g. negative) genomic coordinates are simply not mapped to any sites or probes.

Value

Invisibly, `TRUE` if an existing annotation was replaced and `FALSE` otherwise.

Author(s)

Yassen Assenov

See Also

`rnb.get.annotation` for extracting annotation; `rnb.region.types` for all loaded region types in a genome assembly

Examples

```
my.regions <- data.frame(
    chromosome = c("chr1", "chr1"),
    start = c(49242278L, 49242372L),
    end = c(49242590L, 49242810L),
    rownames = c("BEND5E1", "CpG:38"))
```

```
txt <- "First exon of the BEND5 gene and an overlapping CpG island."
rnb.set.annotation("my regions", my.regions, txt)
```

`rnb.set.annotation.and.cpg.stats`

rnb.set.annotation.and.cpg.stats

Description

wrapper for `rnb.set.annotation` to accept the region format as output by `annotation(rnb.set)`. Additionally, CpG statistics are added to the annotation.

Usage

```
rnb.set.annotation.and.cpg.stats(
  type,
  regions,
  description = NULL,
  assembly = "hg19"
)
```

Arguments

`type`, `description`, `assembly`	Parameters handled exactly as in `rnb.set.annotation`
`regions`	a data.frame handled similarly as by `rnb.set.annotation` with the exception that the genomic location columns should be specified using upper case first letters

Value

Invisibly, `TRUE` if an existing annotation was replaced and `FALSE` otherwise.

Author(s)

Fabian Mueller

See Also

`rnb.set.annotation`

`rnb.show.report` *rnb.show.report*

Description

Opens the given HMTL report file in the browser.

Usage

```
rnb.show.report(report)
```

Arguments

`report`	`Report` object to open.

Value

None (invisible `NULL`).

Author(s)

Pavlo Lutsik

`rnb.step.betadistribution`
rnb.step.betadistribution

Description

Computes the distributions of beta values across various sample groups and adds a corresponding section to the report.

Usage

```
rnb.step.betadistribution(
  rnb.set,
  report,
  columns = rnb.getOption("exploratory.columns"),
  points.per.group = rnb.getOption("distribution.subsample")
)
```

Arguments

`rnb.set`	HumanMethylation450K dataset as an object of type `RnBSet`.
`report`	Report to contain the methylation deviation section. This must be an object of type `Report`.
`columns`	Optional; predefined column names (in the form of a `character` vector) or indices (an `integer` vector) in the sample annotation table. Only these columns are considered for grouping samples and defining profiles. All other columns in the phenotype table are ignored.

`points.per.group`	the targeted number of points (T) per group. Set this to a value < 1 to disable subsampling. More information in the Details section

Value

The modified report.

Details

If subsampling is enabled (i.e. `points.per.group`>0), observations per group are subsampled according to the following procedure: Given K groups and numbers of observed beta values per group N_1,...,N_K, and the target number of points per group T: the total number of points N = sum(N_1,...,N_K) is computed Afterwards the proportions p_k = N_k/N is computed and from each group, S_k = p_k*(K*T) observations are randomly selected from all observations belonging to group k.

Author(s)

Fabian Mueller

`rnb.step.cnv` *rnb.step.cnv*

Description

Performs copy number calling from the Infinium intenstity data and adds the results to the report

Usage

```
rnb.step.cnv(rnb.set, report)
```

Arguments

`rnb.set`	An object of type `RnBeadRawSet`
`report`	Report on quality control to contain the generated sections. This must be an object of type `Report`.

Value

The modified report.

Author(s)

Pavlo Lutsik

`rnb.write.table` *rnb.write.table*

Description

Writes a table to a file. Different formats and compression options are available.

Usage

```
rnb.write.table(tt, fname, fpath = "", format = "csv", gz = FALSE, ...)
```

Arguments

`tt`	Table to be written to file, usually in the form of a `matrix` or `data.frame`.
`fname`	Target file name. If this file already exists, it will be overwritten.
`fpath`	Target file path. If "" (default value), `fname` is assumed to contain the absolute path.
`format`	Target format; one of `"csv"`, `"tab"` or `"txt"`, denoting comma-separated, tab-separated and default text format, respectively. The last format allows for a user-specified delimiter through an additional parameter `sep`. See the documentation of `write.table` for more details.
`gz`	Flag indicating whether the file should be zipped in gz format.
`...`	Any additional arguments to be passed on to `write.table` or `utils::write.csv`.

Value

The (possibly updated) target file name, invisibly. If `gz` is `TRUE`, the string `".gz"` will be appended to `fname`.

Author(s)

Fabian Mueller

See Also

`write.table`

Examples

```
data(mtcars)
rnb.write.table(mtcars,tempfile(pattern="cars",fileext=".csv"))
```

`rnb.xml2options` *rnb.xml2options*

Description

Parses and partially validates parameters and RnBeads options from an XML tree.

Usage

```
rnb.xml2options(fname, return.full.structure = FALSE)
```

Arguments

`fname`	File name containing the XML analysis option values. The name of the root node in this document must be `"rnb.xml"`.
`return.full.structure`	if enabled, return the full structure instead of just the option list

Value

List of two sublists - `"analysis.params"` and `"options"`, storing the specified analysis parameters and previous values of the RnBeads options, respectively.

Author(s)

Yassen Assenov

Examples

```
fname <- paste0("extdata/optionProfiles/",profile,".xml")
rnb.xml2options(system.file(fname,package="RnBeads"))
```

`RnBClusterRun-class` *RnBClusterRun Class*

Description

A class for configuring and running RnBeads on a scientific compute cluster.

Slots

`architecture` A `ClusterArchitecture` object managing the settings for a scientific compute cluster

`modules` A vector of pipeline modules

`module.res.req` Stores the resource requirements for each module. A list containing named vectors for the resources

`module.num.cores` Stores the number of cores for each module

Methods

- `setModuleResourceRequirements,RnBClusterRun,character,character-method` Sets the resource requirements for the different pipeline modules
- `setModuleNumCores,RnBClusterRun,integer,character-method` Sets the number of cores used by the different pipeline modules
- `getModuleNumCores,RnBClusterRun-method` Gets the number of cores used by the different pipeline modules
- `run,RnBClusterRun-method` Submit the pipeline modules to the cluster

Author(s)

Fabian Mueller

`RnBDiffMeth-class` *RnBDiffMeth Class*

Description

A class for storing differential methylation data.

Details

Contains differential methylation tables (DMT) for multiple comparisons and region types. DMTs can be stored in memory as R objects or on disk

Slots

- `sites` List of differential methylation tables on site level (see `computeDiffMeth.bin.site` for details). Indexed by comparison.
- `regions` List of lists of differential methylation tables on region levels (see `computeDiffMeth.bin.region` for details). Indexed by region type on the top level and comparison on the lower level.
- `comparisons` character vector of all comparisons stored in the objects. Vector indices correspond to indices in the `sites` and `regions` list slots.
- `region.types` character vector of all region types stored in the objects. Vector indices correspond to indices in the `regions` list slot.
- `comparison.grouplabels` A character matrix with 2 columns containing group labels of all comparisons in the object
- `comparison.info` A list containing comparison information for each comparison. See `get.comparison.info` for details.
- `includesSites` Logical indicating whether the object contains site-level differential methylation information.
- `site.test.method` method which was applied to obtain the site-level p-values.
- `variability.method` method to be used to detect differentially variable sites.
- `covg.thres` coverage threshold. Important for certain columns of the differential methylation tables.
- `disk.dump` Flag indicating whether the tables should be stored on disk rather than in the main memory
- `disk.path` path on the disk for DMTs.Only meaningful if `disk.dump` is `TRUE`

Methods

- `destroy,RnBDiffMeth-method` remove tables stored to disk from the file system
- `get.region.types,RnBDiffMeth-method` Gets all region types represented in the object as character vector
- `get.comparisons,RnBDiffMeth-method` Gets all comparisons represented in the object as character vector
- `get.comparison.grouplabels,RnBDiffMeth-method` Gets all comparison group names as a matrix
- `get.covg.thres,RnBDiffMeth-method` Gets the coverage threshold employed for obtaining statistics in the differential methylation tables
- `get.table,RnBDiffMeth-method` Gets a differential methylation table
- `addDiffMethTable,RnBDiffMeth-method` Adds a differential methylation table
- `reload,RnBDiffMeth-method` relink disk dumped tables. Useful if the files are manually copied or if the object is loaded again
- `save.tables,RnBDiffMeth-method` save disk dumped tables as binaries and zip them. Useful if the files are copied or shared.
- `join.diffMeth` Merges two disjoint RnBDiffMeth objects into one

Author(s)

Fabian Mueller

`RnBeadClustering-class`

RnBeadClustering Class

Description

Storage class for the results of a clustering algorithm applied on an `RnBSet` dataset.

Slots

- **dissimilarity** Dissimilarity metric used in the form of a one-element `character` vector.
- **dimensionality** Dimensionality of the clustered points in the form of a one-element `integer` vector.
- **algorithm** Clustering algorithm (and optionally, type) as a `character` vector of length 1 or 2.
- **result** Resulting object after applying the clustering algorithm on a dataset.
- **assignments** Cluster assignments for the samples in the dataset as a matrix. Row names in this matrix are sample identifiers, and each column is dedicated to partitioning into k clusters for a fixed k.
- **silhouettes** `numeric` vector of mean silhouette values for each tested value of k.

Methods and Functions

- `samples` Gets the identifiers of all samples used in the clustering.

Author(s)

Yassen Assenov

`RnBeadRawSet-class` *RnBeadRawSet-class*

Description

Main class for storing HumanMethylation micorarray data which includes intensity information

Usage

```
RnBeadRawSet(
  pheno,
  probes,
  M,
  U,
  M0 = NULL,
  U0 = NULL,
  bead.counts.M = NULL,
  bead.counts.U = NULL,
  p.values = NULL,
  qc = NULL,
  platform = "450k",
  beta.offset = 100,
  summarize.bead.counts = TRUE,
  summarize.regions = TRUE,
 region.types = rnb.region.types.for.analysis(ifelse(platform == "MMBC", "mm10",
    ifelse(target == "probesEPICv2", "hg38", rnb.getOption("assembly")))),
  useff = rnb.getOption("disk.dump.big.matrices"),
  ffcleanup = FALSE
)
```

Arguments

`pheno`	Phenotypic data.
`probes`	`character` vector of Infinium(R) probe identifiers
`M`	Matrix of intensities for the probes measuring the abundance of methylated molecules
`U`	Matrix of intensities for the probes measuring the abundance of unmethylated molecules
`M0`	Matrix of "out-of-band" intensities for the probes measuring the abundance of methylated molecules
`U0`	Matrix of "out-of-band" intensities for the probes measuring the abundance of unmethylated molecules
`bead.counts.M`	Matrix of bead counts per probe.
`bead.counts.U`	Matrix of bead counts per probe.
`p.values`	Matrix of detection p-values.
`qc`	...
`platform`	`character` singleton specifying the microarray platform: `"450k"` corresponds to HumanMethylation450 microarray, and `"27k"` stands for HumanMethylation27.

`beta.offset`	A regularization constant which is added to the denominator at beta-value calculation
`summarize.bead.counts`	If `TRUE` the coverage slot is filled by summarizing the `bead.counts.M` and `bead.counts.U` matrices. For type I probes the summarization is done using `min` operation, while for type II probes the bead counts should be identical in both supplied matrices
`summarize.regions`	...
`region.types`	A `character` vector specifying the region types, for which the methylation infromation will be summarized.
`useff`	If `TRUE` the data matrices will be stored as `ff` objects
`ffcleanup`	If `TRUE` and disk dumping has been enabled the data of the input `ff` objects will be deleted

Value

an object of class RnBeadRawSet

Slots

`pheno` Phenotypic data.

`M` `matrix` of intensities for the probes measuring the abundance of methylated molecules.

`U` `matrix` of intensities for the probes measuring the abundance of unmethylated molecules.

`M0` `matrix` of "out-of-band" intensities for the probes measuring the abundance of methylated molecules.

`U0` `matrix` of "out-of-band" intensities for the probes measuring the abundance of unmethylated molecules.

`bead.counts.M` `matrix` of bead counts per probe.

`bead.counts.U` `matrix` of bead counts per probe.

Methods and Functions

`samples` Gets the identifiers of all samples in the dataset.

`M` Get the matrix of intensities for the probes measuring the abundance of methylated molecules.

`U` Get the matrix of intensities for the probes measuring the abundance of unmethylated molecules.

`intensities.by.color` Get probe intensities in each color channel.

Author(s)

Pavlo Lutsik

`RnBeads` *Analysis of genome-scale DNA methylation data with RnBeads*

Description

RnBeads facilitates comprehensive analysis of various types of DNA methylation data at the genome scale. It extends previous approaches for such analysis by high throughput capabilities, as well as presenting results in a comprehensive, highly interpretable fashion.

Details

The complete analysis can be performed by calling the function `rnb.run.analysis`.

Author(s)

Maintainer: Fabian Mueller <`team@rnbeads.org`>

Authors:

- Yassen Assenov <`assenov@gmail.com`>
- Christoph Bock <`cbock@cemm.at`>
- Pavlo Lutsik <`p.lutsik@dkfz.de`>
- Michael Scherer <`mscherer@mpi-inf.mpg.de`>

References

Yassen Assenov*, Fabian Mueller*, Pavlo Lutsik*, Joern Walter, Thomas Lengauer and Christoph Bock (2014) Compehensive Analysis of DNA Methylation Data with RnBeads, Nature Methods, 11(11):1138-1140.

`RnBeads.data` *RnBeads Annotation Tables*

Description

RnBeads uses sets of annotation tables and mappings (from regions to sites) for each of the supported genomes. The structures for one assembly are stored in a separate dedicated annotation package. The following annotation packages are available in Bioconductor:

RnBeads.hg38 for `"hg38"`

RnBeads.hg19 for `"hg19"`

RnBeads.mm10 for `"mm10"`

RnBeads.mm9 for `"mm9"`

RnBeads.rn5 for `"rn5"`

Format

`list` of four elements - `"regions"`, `"sites"`, `"controls"` and `"mappings"`. These elements are described below.

`"regions"` `list` of `NULL`s; the names of the elements correspond to the built-in region annotation tables. Once the default annotations are loaded, the attribute `"builtin"` is a `logical` vector storing, for each region annotation, whether it is the default (built-in) or custom.

`"sites"` `list` of `NULL`s; the names of the elements correspond to the site and probe annotation tables.

`"controls"` `list` of `NULL`s; the names of the elements correspond to the control probe annotation tables. The attribute `"sites"` is a `character` vector pointing to the site annotation that encompasses the respective control probes.

`"mappings"` `list` of `NULL`s; the names of the elements correspond to the built-in region annotation tables.

Details

An assembly-specific scaffold is automatically loaded upon initialization of its annotation, that is, by the first valid call to any of the following functions: `rnb.get.chromosomes`, `rnb.get.annotation`, `rnb.set.annotation`, `rnb.get.mapping`, `rnb.annotation.size`. Adding an annotation amounts to attaching its table(s) and mapping structures to the scaffold.

Author(s)

Yassen Assenov

`RnBeadSet-class` *RnBeadSet Class*

Description

Stores the preprocessed information from HumanMethylation experiments

Usage

```
RnBeadSet(
  pheno,
  probes,
  betas,
  p.values = NULL,
  bead.counts = NULL,
  qc = NULL,
  platform = "450k",
  summarize.regions = TRUE,
 region.types = rnb.region.types.for.analysis(ifelse(platform == "MMBC", "mm10",
    rnb.getOption("assembly"))),
  useff = rnb.getOption("disk.dump.big.matrices")
)
```

Arguments

`pheno`	Phenotypic data.
`probes`	`character` vector of Infinium(R) probe identifiers
`betas`	`matrix` or `ff_matrix` of beta values. If `probes` are missing should contain Infinium probe identifiers as row names.
`p.values`	`matrix` or `ff_matrix` of detection p-values.
`bead.counts`	...
`qc`	...
`platform`	`character` singleton specifying the microarray platform: `"450k"` corresponds to HumanMethylation450 microarray, and `"27k"` stands for HumanMethylation27.
`summarize.regions`	...
`region.types`	A `character` vector specifying the region types, for which the methylation infromation will be summarized.
`useff`	If `TRUE` the data matrices will be stored as `ff` objects

Details

There are multiple ways to create an object of type `RnBeadSet`:

Loading from files Dataset can be loaded from text or binary files. See the function `rnb.execute.import` for more details.

Downloading from GEO See the function `rnb.read.geo` for details.

Converting from `MethyLumiSet` ...

Value

an object of class RnBeadSet

Slots

`pval.sites` `matrix` of detection p-values with the same dimensions as `betas`, or `NULL` if the detection p-values are not available.

`pval.regions` `list` of methylation `matrix` objects, one per available region type. Every row in a matrix corresponds to a methylation site, and every column - to a sample.

`covg.sites` `matrix` of bead counts per probe with the same dimensions as `betas`, or `NULL` if this data are not available.

`qc` Quality control probe information in the form of a `list` of two elements - `"Cy3"` and `"Cy5"`, storing intensities of probes on the green and red channels, respectively. This slot's value is `NULL` if no control probe information is available.

Methods and Functions

`samples` Gets the identifiers of all samples in the dataset.

`pheno` Gets the phenotypic and processing data of the dataset.

`meth` Gets the `matrix` of methylation beta-values of the dataset.

`dpval` Gets the `matrix` of detection p-values of the dataset.

`covg` Gets the `matrix` of bead counts of the dataset.

`qc` Gets the intensities of the quality control probes.

`remove.sites` Removes probes from the dataset.

`remove.samples` Removes samples from the dataset.

`combine` Combines two datasets.

Author(s)

Pavlo Lutsik

`RnBiseqSet-class` *RnBiseqSet Class*

Description

A class for storing the DNA methylation and quality information from bisulfite sequencing experiments

Usage

```
RnBiseqSet(
  pheno,
  sites,
  meth,
  covg = NULL,
  assembly = "hg19",
  target = "CpG",
  summarize.regions = TRUE,
  region.types = rnb.region.types.for.analysis(assembly),
  useff = rnb.getOption("disk.dump.big.matrices"),
  usebigff = rnb.getOption("disk.dump.bigff"),
  verbose = FALSE
)
```

Arguments

`pheno`	phenotypic data.
`sites`	CpG site definition, as a `data.frame` with 3 variables: chromosome (of type `character`), position (`integer`) and strand (`character`, one of "+", "-" or "*"
`meth`	summarized methylation calls as a `matrix` or `ff_matrix`
`covg`	read coverage information as a `matrix` or `ff_matrix`
`assembly`	the genome assembly
`target`	target DNA methylation features (CpG sites)
`summarize.regions`	...
`region.types`	region annotations for which the methylation data should be summarized
`useff`	flag specifying whether the ff functionality should be used
`usebigff`	flag specifying whether the extended ff functionality should be used (large matrix support for ff)
`verbose`	flag specifying whether the diagnostic messages should be written to the console or to the RnBeads logger, if the latter is initialized

Details

TBA

Value

an object of class RnBiseqSet

Slots

`status` Normalization status.

Methods and Functions

`combine` Combines two datasets.

Author(s)

Pavlo Lutsik

`RnBSet-class` *RnBSet Class*

Description

Basic class for storing DNA methylation and experimental quality information

Details

It is a virtual class and objects of type `RnBSet` should not be instantiated. Instead, the child classes are used: `RnBeadRawSet` and `RnBeadSet` for Infinium HumanMethylation and `RnBiseqSet` for bisulfite sequencing data

Slots

`pheno` Sample annotations (phenotypic and processing data) in the form of a `data.frame`.

`sites` A `matrix` object storing the identifiers of the methylation sites for which the methylation information is present

`meth.sites` `matrix` of methylation values. Every row corresponds to a methylation site, and every column - to a sample.

`covg.sites` `matrix` of coverage values. Every row corresponds to a methylation site, and every column - to a sample.

`regions` `list` of all identifiers of methylation sites for which methylation information is available.

`meth.regions` `list` of methylation `matrix` objects, one per available region type. Every row in a matrix corresponds to a methylation site, and every column - to a sample.

`covg.regions` `list` of coverage `matrix` objects, one per available region type. Every row corresponds to a region, and every column - to a sample.

`status` `list` with meta-information about the object.

`assembly` `character` vector of length one, specifying the genome assembly which the object is linked to, e.g. "hg19".

`target` `character` vector of length one, specifying the feature class: `"CpG"` for sequencing data, `"probes450"` and `"probes27"` for HumanMethylation450 and HumanMethylation27 microarrays respectively.

`inferred.covariates` `list` with covariate information. Can contain elements `"sva"` and `"cell.types"`.

`version` Package version in which the dataset was created.

`imputed` Flag indicating if methylation matrix has been imputed.

Methods and Functions

`pheno` Gets the phenotypic and processing data of the dataset.

`samples` Gets the identifiers of all samples in the dataset.

`summarized.regions` Gets the genomic annotations for which methylation data is present.

`meth` Gets a `matrix` of methylation values in the dataset.

`mval` Gets a `matrix` of M values in the dataset.

`covg` Gets the `matrix` of coverage values of the dataset.

`remove.sites` Removes sites from the dataset.

`remove.samples` Removes samples from the dataset.

`addPheno,RnBSet-method` Add sample annotation to the dataset.

`combine` Combines two datasets.

`regionMapping,RnBSet-method` Retrieve the sites mapping to a given region type

`rnb.sample.summary.table` Creates a sample summary table from an RnBSet object.

`isImputed,RnBSet-method` Getter for the imputation slot.

Author(s)

Pavlo Lutsik

`rowOneSampleTP` *rowOneSampleTP*

Description

performs a two-sided t-test for paired samples on each row of a matrix X with the indices inds.1 vs indices inds.g2 as group assignments.

Usage

```
rowOneSampleTP(X, mu = 0, alternative = "two.sided")
```

Arguments

`X`	Matrix on which the test is performed for every row
`mu`	The mean that is tested against
`alternative`	Testing alternative. Must be one of "two.sided" (default),"less","greater" or "all". in case of "all" a data frome with corresping alternative variables is returned. Otherwise the result is a vector.

Value

vector (or data.frame if alternative=="all") of p-values from a paired t-test

Note

Requires `matrixStats` package

Author(s)

Fabian Mueller

Examples

```
library(RnBeads.hg19)
data(small.example.object)
logger.start(fname=NA)
meth.mat <- meth(rnb.set.example)
p.vals <- rowOneSampleTP(meth.mat,mu=0,alternative="greater")
```

`rowPairedTP` *rowPairedTP*

Description

performs a two-sided t-test for paired samples on each row of a matrix X with the indices inds.1 vs indices inds.g2 as group assignments.

Usage

```
rowPairedTP(X, inds.g1, inds.g2 = -inds.g1, alternative = "two.sided")
```

Arguments

`X`	Matrix on which the test is performed for every row
`inds.g1`	column indices of group 1 members. `length(inds.g1)==length(inds.g2)` has to hold true.
`inds.g2`	column indices of group 2 members. `length(inds.g1)==length(inds.g2)` has to hold true.
`alternative`	Testing alternative. Must be one of "two.sided" (default),"less","greater" or "all". in case of "all" a data frome with corresping alternative variables is returned. Otherwise the result is a vector.

Value

vector (or data.frame if alternative=="all") of p-values from a paired t-test

Note

Requires `matrixStats` package

Author(s)

Fabian Mueller

`rowWelchP` *rowWelchP*

Description

performs a two-sided Welch's t-test (unequal variances, unequal sample sizes) on each row of a matrix X with the indices inds.1 vs indices inds.g2 as group assignments.

Usage

```
rowWelchP(
  X,
  inds.g1,
  inds.g2 = -inds.g1,
  na.rm = FALSE,
  alternative = "two.sided"
)
```

Arguments

`X`	Matrix on which the test is performed for every row
`inds.g1`	column indices of group 1 members
`inds.g2`	column indices of group 2 members
`na.rm`	Should NAs be removed (logical)
`alternative`	Testing alternative. Must be one of "two.sided" (default),"less","greater" or "all". in case of "all" a data frome with corresping alternative variables is returned. Otherwise the result is a vector.

Value

vector (or data.frame if alternative=="all") of p-values resulting from the Welch's t-test

Note

Requires `matrixStats` package

Author(s)

Fabian Mueller

Examples

```
library(RnBeads.hg19)
data(small.example.object)
logger.start(fname=NA)
meth.mat <- meth(rnb.set.example)
sample.groups <- rnb.sample.groups(rnb.set.example)[[1]]
p.vals <- rowWelchP(meth.mat,sample.groups[[1]],sample.groups[[2]])
```

`run,RnBClusterRun-method`

run-methods

Description

Runs the analysis by submitting jobs for each module to the compute cluster

Usage

```
## S4 method for signature 'RnBClusterRun'
run(
  object,
  analysis.id,
  config.xml,
  split.differential = TRUE,
  dry.run = FALSE,
  long.cmd.thres = 1024L,
  queue = NULL
)
```

Arguments

`object`	`RnBClusterRun` object
`analysis.id`	analysis id. used for naming submitted jobs and log files
`config.xml`	XML file specifying the analysis options and parameter settings
`split.differential`	flag indicating whether to split the differnetial methylation module into seperate jobs according to sample annotation column and region type.
`dry.run`	Prevent the actual job submission. Rather only write to a shell script file
`long.cmd.thres`	commands that are longer than this number will be encapsulated in shell scripts rather than being submitted as direct command
`queue`	The name of the queue the jobs are going to be submitted to

Value

Nothing of importance

Author(s)

Fabian Mueller

Examples

```
#specify the xml file for your analysis
xml.file <- "MY_ANALYSIS_SETTINGS.XML"
#set the cluster architecture specific to your environment
arch <- new("ClusterArchitectureSGE")
rnb.cr <- new("RnBClusterRun",arch)
#set up the cluster so that 32GB of memory are required (SGE resource is called "mem_free")
```

```
rnb.cr <- setModuleResourceRequirements(rnb.cr,c(mem_free="32G"),"all")
#set up the cluster to use 4 cores on each node for all modules
rnb.cr <- setModuleNumCores(rnb.cr,4L,"all")
#set up the cluster to use 2 cores for the exploratory analysis module
rnb.cr <- setModuleNumCores(rnb.cr,2L,"exploratory")
#run the actual analysis (remove dry.run=TRUE, to really submit the jobs)
run(rnb.cr, "rnbeads_analysis", xml.file, dry.run=TRUE)
```

`run.cross.validation` *run.cross.validation*

Description

This function performs 10-fold cross validation to estimate the performance of a newly trained predictor. If `parallel.isEnabled()`, the function perfoms cross validation in parallel. The function adds a table to the specified `report` containing the result of the 10-fold cross validation.

Usage

```
run.cross.validation(rnbSet, report, alpha = 0.8)
```

Arguments

`rnbSet`	a `RnBSet` object containing the methylation info and ages on which the new predictor should be trained
`report`	report to which the table should be added
`alpha`	alpha parameter used in the elastic net regression

Value

modified report object

Author(s)

Michael Scherer

`sampleCovgApply,RnBSet-method` *sampleCovgApply-methods*

Description

Applies a function over the coverage values for all samples in an `RnBSet` using a low memory footprint.

Usage

```
## S4 method for signature 'RnBSet'
sampleCovgApply(object, fn, type = "sites", ...)
```

Arguments

`object`	object inheriting from `RnBSet`
`fn`	function to be applied
`type`	`character` singleton. Specify "sites" (default) or a region type over which the function is applied
`...`	arguments passed on to the function

Value

Result analogous to `apply(covg(rnbSet, type), 2, FUN=FUN)`

See Also

`covg` Retrieving the matrix of coverage values

`sampleMethApply,RnBSet-method`
sampleMethApply-methods

Description

Applies a function over the methylation values for all samples in an `RnBSet` using a low memory footprint.

Usage

```
## S4 method for signature 'RnBSet'
sampleMethApply(object, fn, type = "sites", ...)
```

Arguments

`object`	object inheriting from `RnBSet`
`fn`	function to be applied
`type`	`character` singleton. Specify "sites" (default) or a region type over which the function is applied
`...`	arguments passed on to the function

Value

Result analogous to `apply(meth(rnbSet, type), 2, FUN=FUN)`

See Also

`meth` Retrieving the matrix of methylation values

`samples,RnBSet-method` *samples-methods*

Description

Extracts sample identifiers

Usage

```
## S4 method for signature 'RnBSet'
samples(object)

## S4 method for signature 'RnBeadClustering'
samples(object)
```

Arguments

`object`	Dataset of interest.

Details

The column of the sample annotation table which contains identifiers is globally controlled via the `"identifiers.column"` option. In case the latter is `NULL` column names of the matrix returned by the `meth` method are treated as sample identifiers. In case the latter are also missing, a `character` vector with sample numbers is returned.

Value

`character` vector of sample identifiers.

Examples

```
library(RnBeads.hg19)
data(small.example.object)
samples(rnb.set.example)
```

`save.rnb.diffmeth` *save.rnb.diffmeth*

Description

save an `RnBDiffMeth` object to disk

Usage

```
save.rnb.diffmeth(object, path)
```

Arguments

`object`	`RnBDiffMeth` object
`path`	path on the disk to save to.

Author(s)

Fabian Mueller

`save.rnb.set` *save.rnb.set*

Description

Consistent saving of an `RnBSet` objects with large matrices of type ff.

Usage

```
save.rnb.set(object, path, archive = TRUE)
```

Arguments

`object`	`RnBSet`-inheriting object.
`path`	the name of the output file (or directory if `archive` is `FALSE`) without an extension. If only the file name is given the object will be saved in the current working directory.
`archive`	if `TRUE` (default value) the output is a ZIP-file.

Details

The saved object can be reloaded with the load.rnb.set function.

Value

invisibly, the full path to the ZIP file (if `archive` is `TRUE`), or to the output directory (otherwise)

Author(s)

Pavlo Lutsik

`save.tables,RnBDiffMeth-method` *save.tables-methods*

Description

save the disk dumped tables to an ff archive for later reloading

Usage

```
## S4 method for signature 'RnBDiffMeth'
save.tables(object, file)
```

Arguments

`object`	`RnBDiffMeth` object
`file`	path on the disk to save to.

Value

success

Author(s)

Fabian Mueller

Examples

```
library(RnBeads.hg19)
data(small.example.object)
logger.start(fname=NA)
pcols <- c("Sample_Group","Treatment")
tdir <- tempfile()
dm <- rnb.execute.computeDiffMeth(rnb.set.example,pcols,disk.dump=TRUE,disk.dump.dir=tdir)
save.tables(dm,tempfile())
```

`set.covariates.ct` *set.covariates.ct*

Description

Adds the results of cell type estimation to an RnBSet

Usage

```
set.covariates.ct(rnb.set, ct.obj)
```

Arguments

`rnb.set`	The `RnBSet` object to which the results should be added
`ct.obj`	An object of class `CellTypeInferenceResult` returned by `rnb.execute.ct.estimation`.

Value

The modified `RnBSet`.

`set.covariates.sva` *set.covariates.sva*

Description

Adds the results of Surrogate Variable Analysis (SVA) to an RnBSet

Usage

```
set.covariates.sva(rnb.set, sva.obj)
```

Arguments

`rnb.set`	The `RnBSet` object to which the results should be added
`sva.obj`	An object of class `SvaResult` as returned by `rnb.execute.sva`.

Value

The modified `RnBSet`. Note that the association information will not be stored.

Author(s)

Fabian Mueller

Examples

```
library(RnBeads.hg19)
data(small.example.object)
logger.start(fname=NA)
sva.obj <- rnb.execute.sva(rnb.set.example,c("Sample_Group","Treatment"),numSVmethod="be")
sva.obj$sva.performed
sva.obj$num.components
rnb.set.mod <- set.covariates.sva(rnb.set.example, sva.obj)
has.covariates.sva(rnb.set.example,"Sample_Group")
has.covariates.sva(rnb.set.mod,"Sample_Group")
```

`setExecutable,ClusterArchitecture,character,character-method` *setExecutable-methods*

Description

Tells the cluster architecture about an executable that can be submitted as job

Usage

```
## S4 method for signature 'ClusterArchitecture,character,character'
setExecutable(object, exec.name, exec.loc)
```

Arguments

`object`	`ClusterArchitecture` object
`exec.name`	A name/identifier that will be associated with the given executable
`exec.loc`	The executable's location

Value

The modified object

Author(s)

Fabian Mueller

`setModuleNumCores,RnBClusterRun,integer,character-method` *setModuleNumCores-methods*

Description

Specifies the number of cores used by the different pipeline modules

Usage

```
## S4 method for signature 'RnBClusterRun,integer,character'
setModuleNumCores(object, num.cores, modules = "all")
```

Arguments

`object`	`RnBClusterRun` object
`num.cores`	an integer specifying the number of cores to be used
`modules`	vector of applicable pipeline modules. Can be `"all"` to specify all modules

Value

The modified object

Author(s)

Fabian Mueller

`setModuleResourceRequirements,RnBClusterRun,character,character-method` *setModuleResourceRequirements-methods*

Description

Specifies resource requirements for the different pipeline modules

Usage

```
## S4 method for signature 'RnBClusterRun,character,character'
setModuleResourceRequirements(object, resources, modules = "all")
```

Arguments

`object`	`RnBClusterRun` object
`resources`	A NAMED character vector containing the resource reuirements as value and the resource name as name
`modules`	vector of applicable pipeline modules. Can be `"all"` to specify all modules

Value

The modified object

Author(s)

Fabian Mueller

`sites,RnBSet-method` *sites-methods*

Description

Methylation sites object information for which is present in the `RnBSet` object.

Usage

```
## S4 method for signature 'RnBSet'
sites(object)
```

Arguments

`object`	Dataset of interest.

Value

A matrix of type `integer` describing the sites, information for which is present in the `object`

Examples

```
library(RnBeads.hg19)
data(small.example.object)
sites(rnb.set.example)
```

---

summarize.regions,RnBSet-method

summarize.regions-methods

Description

Summarize DNA methylation information for which is present in the `RnBSet` object.

Usage

```
## S4 method for signature 'RnBSet'
summarize.regions(
  object,
  region.type,
  aggregation = rnb.getOption("region.aggregation"),
  overwrite = TRUE
)
```

Arguments

`object`	Dataset of interest.
`region.type`	Type of the region annotation for which the summarization will be performed or `"strands"` for summarizing the methylation values from both strands
`aggregation`	Operation to summarize the methylation values. Currently supported values are `"mean"`, `"median"`, `"min"`, `"max"` and `"coverage.weighted"`
`overwrite`	If `TRUE` the existing region-level information for `region.type` is discarded

Value

object of the same class as the supplied one containing the summarized methylation information for the specified region types

Examples

```
library(RnBeads.hg19)
data(small.example.object)
rnb.set.summarized<-summarize.regions(rnb.set.example, "genes", overwrite=TRUE)
head(meth(rnb.set.summarized, type="genes", row.names=TRUE))
```

`summarized.regions,RnBSet-method` *summarized.regions-methods*

Description

Gets the genomic annotations for which methylation data is present in the `RnBSet` object.

Usage

```
## S4 method for signature 'RnBSet'
summarized.regions(object)
```

Arguments

`object`	Methylation dataset of interest.

Value

`character` vector listing all genomic annotations summarized in the given dataset. If the dataset contains methylation in sites only, an empty vector is returned.

Author(s)

Yassen Assenov

See Also

`summarize.regions` for calculating region-wise methylation in a dataset; `rnb.set.annotation` for adding or replacing a region annotation table

Examples

```
library(RnBeads.hg19)
data(small.example.object)
summarized.regions(rnb.set.example)
```

`U,RnBeadRawSet-method` *U-methods*

Description

Extract raw unmethylated probe intensity from an object of `RnBeadRawSet` class.

Usage

```
## S4 method for signature 'RnBeadRawSet'
U(object, row.names = FALSE)
```

Arguments

`object`	Dataset of interest.
`row.names`	Flag indicating whether the resulting matrix will be assigned row names

Value

`matrix` of the unmethylated probe intensities

Examples

```
library(RnBeads.hg19)
data(small.example.object)
U.intensity<-U(rnb.set.example)
head(U.intensity)
```

`updateMethylationSites,RnBSet-method` *updateMethylationSites-methods*

Description

Replaces the methylation info with the specified data frame.

Usage

```
## S4 method for signature 'RnBSet'
updateMethylationSites(object, meth.data, verbose = FALSE)
```

Arguments

`object`	Dataset of interest.
`meth.data`	This object has to be a `data.frame` of equal dimension than the one already contained in `object`, containing the methylation info that should be associated with the object.
`verbose`	if `TRUE` additional diagnostic output is generated

Value

The modified dataset. #’

`updateRegionSummaries,RnBSet-method`
updateRegionSummaries

Description

Updates the region information present in an RnBSet by invoking summarize.regions on all region types present in the object

Usage

```
## S4 method for signature 'RnBSet'
updateRegionSummaries(object)
```

Arguments

`object`	Dataset of interest.

Value

Sample annotation information available for the dataset in the form of a `data.frame`.

`[,BigFfMat,ANY,ANY,ANY-method`
Extract parts of BigFfMat

Description

Extract parts of BigFfMat

Usage

```
## S4 method for signature 'BigFfMat,ANY,ANY,ANY'
x[i, j, drop = TRUE]
```

Arguments

`x`	BigFfMat object
`i`	row indices (integer, logical, character are allowed)
`j`	column indices (integer, logical, character are allowed)
`drop`	analogous to generic drop

`[<-,BigFfMat,ANY,ANY,ANY-method`

Replace parts of BigFfMat

Description

Replace parts of BigFfMat

Usage

```
## S4 replacement method for signature 'BigFfMat,ANY,ANY,ANY'
x[i, j] <- value
```

Arguments

`x`	BigFfMat object
`i`	row indices (integer, logical, character are allowed)
`j`	column indices (integer, logical, character are allowed)
`value`	replacement values

Index