Package 'MEAL'

October 27, 2015

Title Perform methylation analysis

Version 1.0.1

Description Package to integrate methylation and expression data. It can also perform methylation or expression analysis alone. Several plotting functionalities are included as well as a new region analysis based on redundancy analysis. Effect of SNPs on a region can also be estimated.

Depends R (>= 3.2.0), Biobase

License Artistic-2.0

biocViews DNAMethylation, Microarray, Software, WholeGenome

LazyData true

Imports GenomicRanges, SNPassoc, limma, DMRcate, snpStats, vegan, BiocGenerics, minfi, IRanges, S4Vectors, methods, doParallel, parallel, ggplot2, sva, scales

Suggests testthat, IlluminaHumanMethylation450kanno.ilmn12.hg19, knitr, minfiData, MEALData, BiocStyle

VignetteBuilder knitr

NeedsCompilation no

Author Carlos Ruiz [aut, cre], Carles Hernandez-Ferrer [aut], Juan R. Gonz<c3><a1>lez [aut]

Maintainer Carlos Ruiz <cruiz@creal.cat>

R topics documented:

`add.genexp` *Method to add a slot of expression to* `MultiDataSet`.

Description

This method adds or overwrites the slot `"expression"` of an `MultiDataSet` with the content of the given `ExpressionSet`.

Usage

```
add.genexp(object, gexpSet, warnings = TRUE)
```

Arguments

`object`	`MultiDataSet` that will be filled.
`gexpSet`	`ExpressionSet` to be used to fill the slot.
`warnings`	Logical to indicate if warnings will be displayed.

Value

A new `MultiDataSet` with the slot `"expression"` filled.

Examples

```
multi <- new("MultiDataSet")
eset <- new("ExpressionSet", exprs = matrix(runif(4), 2))
fData(eset) <- data.frame(chromosome = c("chr1", "chr2"), start = c(12414, 1234321),
 end = c(121241, 12412412414), stringsAsFactors = FALSE)
multi <- add.genexp(multi, eset)
```

---

add.methy *Method to add a slot of methylation to* MultiDataSet.

Description

This method adds or overwrites the slot "methylation" of an MultiDataSet with the content of the given MethylationSet.

Usage

```
add.methy(object, methySet, warnings = TRUE)
```

Arguments

object	MultiDataSet that will be filled.
methySet	MethylationSet to be used to fill the slot.
warnings	Logical to indicate if warnings will be displayed.

Value

A new MultiDataSet with the slot "methylation" filled.

Examples

```
if (require(MEALData)){
 multi <- new("MultiDataSet")
 betavals <- betavals[1:100, ]  ## To speed up the example, the beta values are reduced
 methy <- prepareMethylationSet(betavals, pheno)
 multi <- add.methy(multi, methy)
}
```

---

add.set *Method to add a slot to* MultiDataSet.

Description

This method adds or overwrites a slot of a MultiDataSet with the content of the given eSet.

Usage

```
add.set(object, set, dataset.name, warnings = TRUE)
```

Arguments

object	MultiDataSet that will be filled.
set	Object derived from eSet to be used to fill the slot.
dataset.name	Character with the name of the slot to be filled.
warnings	Logical to indicate if warnings will be displayed.

Value

A new MultiDataSet with a slot filled.

Examples

```
multi <- new("MultiDataSet")
eset <- new("ExpressionSet", exprs = matrix(runif(10), 5))
multi <- add.set(multi, eset, "exampledata")
```

---

add.snps *Method to add a slot of SNPs to* MultiDataSet.

Description

This method adds or overwrites the slot "snps" of an MultiDataSet with the content of the given SnpSet.

Usage

```
add.snps(object, snpSet, warnings = TRUE)
```

Arguments

object	MultiDataSet that will be filled.
snpSet	SnpSet to be used to fill the slot.
warnings	Logical to indicate if warnings will be displayed.

Value

A new MultiDataSet with the slot "snps" filled.

AnalysisRegionResults *AnalysisRegionResults instances*

Description

AnalysisResults heir with the analyses performed in a range of the whole genome.

AnalysisRegionResults instances

Usage

```
analysisRegionResults(analysisResults, set, range, snpspvals = data.frame(),
  regionlm = list(), relevantsnps = character(), snpsVar = as.numeric(NA),
  equation = NULL)

## S4 method for signature 'AnalysisRegionResults'
getRange(object)

## S4 method for signature 'AnalysisRegionResults'
getRDA(object)

## S4 method for signature 'AnalysisRegionResults'
regionLM(object)

## S4 method for signature 'AnalysisRegionResults'
regionPval(object)

## S4 method for signature 'AnalysisRegionResults'
regionR2(object)

## S4 method for signature 'AnalysisRegionResults'
snps(object)

## S4 method for signature 'AnalysisRegionResults'
snpsPvals(object)

## S4 method for signature 'AnalysisRegionResults'
snpsVar(object)

## S4 method for signature 'AnalysisRegionResults'
plotRDA(object, n_feat = 5)

## S4 method for signature 'AnalysisRegionResults'
plotRegionR2(object, feat, ...)
```

Arguments

analysisResults	AnalysisResults
set	MethylationSet or ExpressionSet
range	GenomicRanges

snpspvals	Data.frame obtained from calculateRelevantSNPs
regionlm	Data.frame obtained from explainedVariance
relevantsnps	Character vector with the relevant snps names
snpsVar	Numeric with the variability of the SNP matrix explained by the components used to adjust the linear model.
equation	Character containing the formula to be used to create the model.
object	MethylationResults
n_feat	Numeric with the number of features to be highlighted.
feat	Numeric with the index of the cpg or character with its name.
...	Further arguments passed to plotLM

Value

An AnalysisRegionResults

Methods (by generic)

- getRange: Get range where the analyses was performed
- getRDA: Get rda object.
- regionLM: Get R2 values of cpgs vs variables.
- regionPval: Get p-value of lineal model R2.
- regionR2: Get R2 of the region vs variables lineal model
- snps: Get SNPs data
- snpsPvals: Get p-values of correlations of snps-cpgs pairs
- snpsVar: Get variance of SNP matrix present in the component used to adjusting.
- plotRDA: Plot RDA results
- plotRegionR2: Plot R2 region values

Slots

range GenomicRanges used to perform the analysis.

snps Character vector with the snps that are correlated to at least one cpg.

snpsPvals Data.frame with the results of the correlation test SNP-cpg.

snpsVar Numeric with the variability of the SNP matrix explained by the components used to adjust the linear model.

rda rda object from vegan package with the results of RDA analysis in the range.

regionLM List with the R2 of the linear model of beta values against our variable of interest and against significant SNPs for each cpg.

regionR2 Numeric with the R2 of the region calculated using a redundancy analysis.

regionPval Numeric with the pval of the region's R2.

Examples

```
showClass("AnalysisRegionResults")
```

AnalysisResults *AnalysisResults instances*

Description

Container with the results of per probe and per region analyses.

AnalysisResults instances

Usage

```
analysisResults(set, model, regionResults, probeResults, num_feat = 50,
  num_vars = ncol(pData(set)))

## S4 method for signature 'AnalysisResults'
blocks(object)

## S4 method for signature 'AnalysisResults'
bumps(object)

## S4 method for signature 'AnalysisResults'
covariableNames(object)

## S4 method for signature 'AnalysisResults'
dmrCate(object)

## S4 method for signature 'AnalysisResults'
feats(object)

## S4 method for signature 'AnalysisResults'
featvals(object)

## S4 method for signature 'AnalysisResults'
getGeneVals(object, gene)

## S4 method for signature 'AnalysisResults'
getMs(object, threshold = 1e-04)

## S4 method for signature 'AnalysisResults'
model(object)

## S4 method for signature 'AnalysisResults'
modelVariables(object)

## S4 method for signature 'AnalysisResults'
phenoData(object)

## S4 replacement method for signature 'AnalysisResults,ANY'
phenoData(object) <- value

## S4 method for signature 'AnalysisResults'
```

```
pData(object)

## S4 replacement method for signature 'AnalysisResults,ANY'
pData(object) <- value

## S4 method for signature 'AnalysisResults'
probeResults(object)

## S4 method for signature 'AnalysisResults'
regionResults(object)

## S4 method for signature 'AnalysisResults'
sampleNames(object)

## S4 method for signature 'AnalysisResults'
variableNames(object)

## S4 method for signature 'AnalysisResults'
exportResults(object, dir = "./", prefix = NULL,
  vars = modelVariables(object))

## S4 method for signature 'AnalysisResults'
plotEWAS(object,
  variable = modelVariables(object)[1], range = NULL)

## S4 method for signature 'AnalysisResults'
plotQQ(object,
  variable = modelVariables(object)[1])

## S4 method for signature 'AnalysisResults'
plotRegion(object,
  variable = modelVariables(object)[[1]], range = NULL)

## S4 method for signature 'AnalysisResults'
plotVolcano(object,
  variable = modelVariables(object)[1])
```

Arguments

set	MethylationSet or ExpressionSet used to perform the analysis
model	Model matrix used to produce the calculations
regionResults	List with the region results
probeResults	List with the probe results
num_feat	Numeric with the minimum number of feature values to be included.
num_vars	Numeric with the number of columns of the pData table that should be considered as variables.
object	AnalysisResults
gene	Character with the name of the gene
threshold	Numeric with the threshold to avoid 0s and 1s.
value	AnnotatedDataFrame or data.frame with the phenotype

dir	Character with the path to export.
prefix	Character with a prefix to be added to all file names.
vars	Character vector with the names of the variables to be exported. Note: names should be that of the model.
variable	Character with the variable name used to obtain the probe results. Note: model name should be used. Original variable name might not be valid.
range	GenomicRange whose probes will be highlighted

Value

AnalysisResults

Methods (by generic)

- blocks: Get BlockFinder analysis results
- bumps: Get Bumphunter analysis results
- covariableNames: Get covariable names
- dmrCate: Get dmrCate analysis results
- feats: Get features names
- featvals: Get features values matrix
- getGeneVals: Get probe results of a gene
- getMs: Get Ms values
- model: Get model used to perform the analysis
- modelVariables: Get names of the variables in the model matrix
- phenoData: Get phenotypes data (AnnotatedDataFrame)
- phenoData<-: Set phenotypes data (AnnotatedDataFrame)
- pData: Get phenotypes data (data.frame)
- pData<-: Set phenotypes data (data.frame)
- probeResults: Get per probe analysis results
- regionResults: Get all per region analysis results
- sampleNames: Get sample names
- variableNames: Get variable names
- exportResults: Exports results data.frames to csv files.
- plotEWAS: Plot a Manhattan plot with the probe results
- plotQQ: QQ plot of probe analysis
- plotRegion: Plot of the region
- plotVolcano: Make a Volcano plot with the probe results

Slots

`originalclass` Character with the class of the object used to perform the analysis

`features` Matrix with the values of the most significant features.

`phenotypes` AnnotatedDataFrame with the phenotypes.

`model` Matrix with the model used in the analysis

`sampleNames` Character vector with the names of the samples

`variableNames` Character vector with the names of the variables used in the analysis. Names are equal to those find in phenotypes.

`covariableNames` Character vector with the names of the covariables used in the analysis. Names are equal to those find in phenotypes.

`results` List of data.frames with the results of per probe analysis. Names are those of the model.

`DMRcate` List of data.frames with the results of `DMRcate`. Names are those of the model.

`Bumphunter` List of data.frames with the results of `Bumphunter`. Names are those of the model.

`BlockFinder` List of data.frames with the results of `BlockFinder`. Names are those of the model.

Examples

```
showClass("AnalysisResults")
```

`calculateRelevantSNPs` *Calculate the SNPs correlated to cpgs*

Description

This function estimates the correlation between the snps and the cpgs. For each pair cpg-SNP the p-value is returned.

Usage

```
calculateRelevantSNPs(set, snps, num_cores = 1)
```

Arguments

`set`	`MethylationSet`
`snps`	`SnpSet`
`num_cores`	Numeric with the number of cores to be used.

Value

Data.frame with the pvalues for pairs SNPs-cpgs. SNPs are in the rows and cpgs in the columns.

Examples

```
## Not run:
## betamatrix: matrix of beta values
## phenodf: data.frame with the phenotypes
## snpsobject: SnpSet
set <- prepareMethylationSet(matrix = betamatrix, phenotypes = phenodf)
relevantSNPs <- calculateRelevantSNPs(set, snpsobject)

## End(Not run)
```

checkProbes *Filter* MethylationSet *probes*

Description

This function selects probes present in the annotation matrix. Probes without annotation and annotation values without beta values are discarded.

Usage

```
checkProbes(object)
```

Arguments

object	MethylationSet

Value

MethylationSet containing the common samples.

Examples

```
if (require(MEALData)){
 betavals <- betavals[1:100, ]  ## To speed up the example, the beta values are reduced
 methy <- prepareMethylationSet(betavals, pheno)
 checkProbes(methy)
}
```

checkSamples *Modify a* MethylationSet *to only contain common samples*

Description

This function removes samples that have beta values but no phenotypes and vice versa. If snps object is present, only samples present in the three set are retained.

Usage

```
checkSamples(object)
```

Arguments

object	MethylationSet

Value

MethylationSet containing the common samples.

Examples

```
if (require(MEALData)){
 betavals <- betavals[1:100, ]  ## To speed up the example, the beta values are reduced
 methy <- prepareMethylationSet(betavals, pheno)
 checkSamples(methy)
}
```

`chrNumToChar` *Convert chr numbers to chr strings*

Description

Given a vector of number representing the chromosomes, convert them to string (e.g 1 to chr1). 23 is consider chrX, 24 is chrY, 25 is chrXY (probes shared between chromosomes X and Y) and 26 is chrMT.

Usage

```
chrNumToChar(vector)
```

Arguments

`vector`	The vector with the chromosome numbers

Value

A vector with the chromosomes in string format.

Examples

```
chromosomes <- c(1, 3, 4, 23, 15)
stringChrs <- chrNumToChar(chromosomes)
stringChrs
```

`correlationMethExprs` *Computes the correlation between methylation and expression*

Description

Estimates the correlation between methylation and expression. When there are known variables that affect methylation and/or expression, their effect can be substracted using a linear model and then the residuals are used.

Usage

```
correlationMethExprs(multiset, vars_meth = NULL, vars_exprs = NULL,
  vars_meth_types = rep(NA, length(vars_meth)), vars_exprs_types = rep(NA,
  length(vars_exprs)), flank = 250000, num_cores = 1, verbose = TRUE)
```

Arguments

multiset	MultiDataSet containing a methylation and an expression slots.
vars_meth	Character vector with the names of the variables that will be used to obtain the methylation residuals. By default, none is used and residuals are not computed.
vars_exprs	Character vector with the names of the variables that will be used to obtain the expression residuals. By default, none is used and residuals are not computed.
vars_meth_types	Character vector with the types of the methylation variables. By default, variables type won't be changed.
vars_exprs_types	Character vector with the types of the expression variables. By default, variables type won't be changed.
flank	Numeric with the number of pair bases used to define the cpg-expression probe pairs.
num_cores	Numeric with the number of cores to be used.
verbose	Logical value. If TRUE, it writes out some messages indicating progress. If FALSE nothing should be printed.

Details

For each cpg, a range is defined by the position of the cpg plus the flank parameter (upstream and downstream). Only those expression probes that are entirely in this range will be selected. For these reason, it is required that the ExpressionSet contains a featureData with the chromosome and the starting and ending positions of the probes.

Value

Data.frame with the results of the linear regression:

- cpg: Name of the cpg
- exprs: Name of the expression probe
- beta: coefficient of the methylation change
- se: standard error of the beta
- P.Value: p-value of the beta coefficient
- adj.P.Val: q-value computed using B&H

---

createRanges *Create* GenomicRanges *from data.frame*

Description

Convert a data.frame with chromosomes in the first column, starting positions in the second one and ending position in the third one to GenomicRanges. Names of the data.frame are preserved in the output GenomicRanges.

Usage

```
createRanges(ranges)
```

Arguments

ranges	Data.frame or matrix

Value

GenomicRanges

Examples

```
dfranges <- data.frame(chr = c("chr1", "chr2", "chr1"), start = c(1290, 1250, 4758),
end = c(64389, 632409, 16430), stringsAsFactors = FALSE)
names(dfranges) <- c("range1", "range2", "range3")
ranges <- createRanges(dfranges)
ranges
```

DAPipeline *Perform differential methylation analysis*

Description

Wrapper for analysing differential methylation and expression at region and probe level.

Usage

```
DAPipeline(set, variable_names, variable_types = rep(NA,
  length(variable_names)), covariable_names = NULL,
  covariable_types = rep(NA, length(covariable_names)), equation = NULL,
  num_var = NULL, labels = NULL, sva = FALSE,
  region_methods = c("bumphunter", "DMRcate"), shrinkVar = FALSE,
  probe_method = "robust", max_iterations = 100, num_feat = 50,
  num_cores = 1, verbose = FALSE, ...)
```

Arguments

set	MethylationSet or ExpressionSet
variable_names	Character vector with the names of the variables that will be returned as result.
variable_types	Character vector with the types of the variables. As default, variables type won't be changed.
covariable_names	Character vector with the names of the variables that will be used to adjust the model.
covariable_types	Character vector with the types of the covariables. As default, variables type won't be changed.
equation	Character containing the formula to be used to create the model.
num_var	Numeric with the number of variables in the matrix for which the analysis will be performed. Compulsory if equation is not null.
labels	Character vector with the labels of the variables.
sva	Logical indicating if Surrogate Variable Analysis should be applied.

region_methods	Character vector with the methods used in DARegion. If "none", region analysis is not performed.
shrinkVar	Logical indicating if shrinkage of variance should be applied in probe analysis.
probe_method	Character with the type of linear regression applied in probe analysis ("ls" or "robust")
max_iterations	Numeric with the maximum of iterations in the robust regression.
num_feat	Numeric with the minimum number of cpg beta values to be included in the results.
num_cores	Numeric with the number of cores to be used.
verbose	Logical value. If TRUE, it writes out some messages indicating progress. If FALSE nothing should be printed.
...	Further arguments passsed to DARegion function.

Details

This function is the main wrapper of the package. First, it simplifies the the set to only contain the common samples between phenotype and features. In addition, it allows to change the class of the variables and to apply genomic models (more information on preparePhenotype). Afterwards, analysis per probe and per region are done merging the results in an AnalysisResults object.

Default linear model will contain a sum of the variables and covariables. If interactions are desired, a costum formula can be specified. In that case, variables and covariables must also be specified in order to assure the proper work of the resulting AnalysisResult. In addition, the number of variables of the model for which the calculation will be done **must** be specified.

Value

MethylationResult object

See Also

preparePhenotype

Examples

```
if (require(minfiData)){
 set <- prepareMethylationSet(matrix = getBeta(MsetEx)[1:10, ], pheno = pData(MsetEx))
 res <- DAPipeline(set, variable_names = "Sample_Group", probe_method = "ls")
 res
}
```

DAProbe *Perform per probe analysis*

Description

Compute statistics (t estimate and p-value) for methylation or expression data using linear or robust linear regression.

Usage

```
DAProbe(set, model, coefficient = 2, shrinkVar = FALSE, method = "robust",
  max_iterations = 100)
```

Arguments

set	MethylationSet, matrix of M-values or ExpressionSet.
model	Matrix with the model
coefficient	Numeric with the index of the model matrix used to perform the analysis. If a vector is supplied, a list will be returned.
shrinkVar	Logical indicating if shrinkange of variance should be performed.
method	String indicating the method used in the regression ("ls" or "robust")
max_iterations	Numeric indicating the maximum number of iterations done in the robust method.

Value

Data.frame or list of data.frames containing intercept and slope values. If the set is a MethylationSet, probe's position, chromosome and the nearest gene are also returned.

Examples

```
if (require(minfiData)){
 mvalues <- getM(MsetEx)[1:100, ]
 model <- model.matrix(~ Sample_Group, data = pData(MsetEx))
 res <- DAProbe(mvalues, model, method = "ls")
 head(res)
}
```

DARegion *Detect regions differentially methylated*

Description

This function is a wrapper of two known region differentially methylated detection methods: *Bumphunter* and *DMRcate*. blockFinder implementation present in minfi package is also available.

Usage

```
DARegion(set, model, proberes, methods = c("blockFinder", "bumphunter",
  "DMRcate"), coefficient = 2, num_permutations = 0,
  bumphunter_cutoff = 0.05, bumps_max = 30000, num_cores = 1,
  verbose = FALSE, ...)
```

Arguments

set	MethylationSet.
model	Model matrix representing a linear model.
proberes	Data.frame or list of data.frames with the results of DAProbe
methods	Character vector with the names of the methods used to estimate the regions. Valid names are: "blockFinder", "bumphunter" and "DMRcate".

coefficient	Numeric with the index of the model matrix used to perform the analysis.
num_permutations	Numeric with the number of permutations used to calculate p-values in bumphunter and blockFinder
bumphunter_cutoff	Numeric with the threshold to consider a probe significant. If one number is supplied, the lower limit is minus the upper one. If two values are given, they will be upper and lower limits.
bumps_max	Numeric with the maximum number of bumps allowed.
num_cores	Numeric with the number of cores used to perform the permutation.
verbose	Logical value. If TRUE, it writes out some messages indicating progress. If FALSE nothing should be printed.
...	Further arguments passsed to bumphunter function.

Details

DARegion performs a methylation region analysis using *bumphunter* and *DMRcate*. Bumphunter allows the modification of several parameters that should be properly used.

Cutoff will determine the number of bumps that will be detected. The smaller the cutoff, the higher the number of positions above the limits, so there will be more regions and they will be greater. Bumphunter can pick a cutoff using the null distribution, i.e. permutating the samples. There is no standard cutoff and it will depend on the features of the experiment. Permutations are used to estimate p-values and, if needed, can be used to pick a cutoff. The advised number of permutation is 1000. The number of permutations will define the maximum number of bumps that will be considered for analysing. The more bumps, the longer permutation time. As before, there is not an accepted limit but minfi tutorial recommends not to exceed 30000 bumps. Finally, if supported, it is very advisable to use parallelization to perform the permutations.

Due to minfi design, *BlockFinder* can only be run using own minfi annotation. This annotation is based on hg19 and Illumina 450k chipset. Cpg sites not named like in this annotation package will not be included. As a result, the use of *BlockFinder* is not recommended.

DMRcate uses a first step where linear regression is performed in order to estimate coefficients of the variable of interest. This first step is equal to the calculation performed in DAProbe, but using in this situation linear regression and not robust linear regression. The results of DAProbe can be supplied in proberes argument, skipping this first step.

DARegion supports multiple variable analyses. If coefficient is a vector, a list of lists will be returned. Each member will be named after the name of the column of the model matrix.

Value

List with the main results of the three methods. If a method is not chosen, NA is returned in this position.

See Also

bumphunter, blockFinder, dmrcate

Examples

```
if (require(minfiData)){
 set <- prepareMethylationSet(minfi::getBeta(MsetEx)[1:10, ], pheno = pData(MsetEx))
 model <- model.matrix(~Sample_Group, data = pData(MsetEx))
```

```
 res <- DARegion(set, model)
 res
}
```

DARegionAnalysis *Analyse methylation or expression in a specific range*

Description

Methylation analysis in a genomic range, taking into account snps.

Usage

```
DARegionAnalysis(set, range, omicset = "methylation", variable_names,
  variable_types = rep(NA, length(variable_names)), covariable_names = NULL,
  covariable_types = rep(NA, length(covariable_names)), equation = NULL,
  num_var = NULL, labels = NULL, sva = FALSE, use_snps = TRUE,
  snps_cutoff = 0.01, region_methods = c("blockFinder", "bumphunter",
  "DMRcate"), shrinkVar = FALSE, probe_method = "robust",
  max_iterations = 100, num_cores = 1, verbose = FALSE, ...)
```

Arguments

set	MethylationSet, ExpressionSet or MultiDataSet.
range	GenomicRanges with the desired range.
omicset	In a MultiDataSet allows to choose between methylation and expression (valid values are: "methylation" or "expression").
variable_names	Character vector with the names of the variables that will be returned as result.
variable_types	Character vector with the types of the variables. By default, variables type won't be changed.
covariable_names	Character vector with the names of the variables that will be used to adjust the model.
covariable_types	Character vector with the types of the covariables. By default, variables type won't be changed.
equation	String containing the formula to be used to create the model.
num_var	Numeric with the number of variables in the matrix for which the analysis will be performed. Compulsory if equation is not null.
labels	Character vector with the labels of the variables.
sva	Logical indicating if Surrogate Variable Analysis should be applied.
use_snps	Logical indicating if SNPs should be used in the analysis.
snps_cutoff	Numerical with the threshold to consider a SNP-cpg correlation p-value significant.
region_methods	Character vector with the methods used in DARegion. If "none", region analysis is not performed.
shrinkVar	Logical indicating if shrinkage of variance should be applied in probe analysis.

probe_method	Character with the type of linear regression applied in probe analysis ("ls" or "robust")
max_iterations	Numeric with the maximum of iterations in the robust regression.
num_cores	Numeric with the number of cores to be used.
verbose	Logical value. If TRUE, it writes out some messages indicating progress. If FALSE nothing should be printed.
...	Further arguments passsed to DAPipeline function.

Details

Set is filtered to the range specified. If SNPs are present in the set, those are also filtered and then, correlation between SNPs and cpgs is tested. SNPs that are correlated to at least one cpg are added to covariables. After that, DAPipeline is run. RDA test of the region is performed, returning the R2 between the variables and the beta matrix and a p-value of this R2.

Value

AnalysisRegionResult object

See Also

preparePhenotype, DAPipeline

Examples

```
if (require(minfiData)){
 set <- prepareMethylationSet(getBeta(MsetEx)[1:1000, ], pheno = pData(MsetEx))
 range <- GenomicRanges::GRanges(seqnames=Rle("chrX"),
 ranges = IRanges(30000, end = 123000000))
 res <- DARegionAnalysis(set, range = range, variable_names = "Sample_Group",
 probe_method = "ls")
 res
}
```

---

explainedVariance *Calculate R2 for different variables*

Description

Using a data.frame as input, calculates the R2 between a dependent variable and some independent variables. Base adjusting by covariates can also be used.

Usage

```
explainedVariance(data, num_mainvar = 1, num_covariates = 0,
  variable_label = NULL)
```

Arguments

data	Data.frame containing the dependent variable in the first column.
num_mainvar	Numerical with the number of variables that should be grouped. They should be at the beggining.
num_covariates	Numerical with the number of variables that should be considered as covariates. Covariates variables must be at the end.
variable_label	Character with the name of the main variable in the results.

Details

explainedVariance computes R2 via linear models. The first column is considered to be the dependent variable. Therefore, a lineal model will be constructed for each of the remaining variables. In case that covariates were included, they will be included in all the models and, in addition, a model containing only the covariates will be returned.

Some variables can be grouped in the models to assess their effect together.

Value

Numeric vector with the R2 explained by each of the variables.

Examples

```
data(mtcars)
R2 <- explainedVariance(mtcars)
R2
```

---

exportResults *Exports results data.frames to csv files.*

Description

Exports results to csv files. If more than one variable is present, subfolders with the name of the variable are created. For each variable, four files will be generated: probeResults.csv, dmrCateResults.csv, bumphunterResults.csv and blockFinderResults.csv

Usage

```
exportResults(object, dir = "./", prefix = NULL,
  vars = modelVariables(object))
```

Arguments

object	MethylationResults or MethylationRegionResults
dir	Character with the path to export.
prefix	Character with a prefix to be added to all file names.
vars	Character vector with the names of the variables to be exported. Note: names should be that of the model.

Value

Files are saved into the given folder.

Examples

```
if (require(minfiData)){
set <- prepareMethylationSet(getBeta(MsetEx)[1:10,], pheno = pData(MsetEx))
methyOneVar <- DAPipeline(set, variable_names = "sex", probe_method = "ls")
exportResults(methyOneVar)
}
```

---

filterSet *Filter a* MethylationSet*, an* ExpressionSet *or a* SnpSet

---

Description

Filter a MethylationSet, an ExpressionSet or a SnpSet

Usage

```
filterSet(set, range)
```

Arguments

set	MethylationSet, ExpressionSet or a SnpSet
range	GenomicRanges with the desired range.

Value

MethylationSet, ExpressionSet or a SnpSet with only the features of the range.

Examples

```
if (require(minfiData) & require(GenomicRanges)){
range <- GRanges(seqnames=Rle("chrY"),
ranges = IRanges(3000000, end=12300000))
set <- prepareMethylationSet(MsetEx[1:100, ], pData(MsetEx))
set
filteredset <- filterSet(set, range)
filteredset
}
```

getGeneVals *Get all probes related to gene*

Description

Given a MethylationResults and a gene name returns the results of the analysis of all the probes of the gene.

Usage

```
getGeneVals(object, gene)
```

Arguments

object	MethylationResults
gene	Character with the name of the gene

Value

List of data.frames with the results of the analysis of the probes belonging to the gene

Examples

```
if (require(minfiData)){
set <- prepareMethylationSet(getBeta(MsetEx)[1:10,], pheno = pData(MsetEx))
methyOneVar <- DAPipeline(set, variable_names = "sex", probe_method = "ls")
getGeneVals(methyOneVar, "TSPY4")
}
```

getMs *Transforms beta values to M-values*

Description

Given a MethylationSet or a AnalysisResults returns the matrix of M values using a logit2 transformation. Betas equal to 0 will be transformed to threshold and betas equal to 1, to 1 - threshold.

Usage

```
getMs(object, threshold = 1e-04)
```

Arguments

object	MethylationSet or AnalysisResults
threshold	Numeric with the threshold to avoid 0s and 1s.

Value

Matrix with the M values.

Examples

```
if (require(minfiData)){
set <- prepareMethylationSet(MsetEx[1:100, ], pData(MsetEx))
mvalues <- getMs(set)
head(mvalues)
}
```

MEAL *MEAL (Methylation and Expression AnaLizer): Package for analysing methylation and expression data*

Description

MEAL has three different categories of important functions: processing, analysing and plotting.

processing

Functions used to create MEAL objects and to modify them. Main functions are prepareMethylationSet and preparePhenotype

analysing

Functions used to perform the analysis of methylation data. DAProbe performs per probe analysis and DARegion performs per region analysis. There are two wrappers: DAPipeline and DARegionAnalysis that performs per probe and per region analysis. The first one analyses the whole methylation sites and the second one only a given region. Finally, correlationMethExprs and multiCorrMethExprs compute the correlation between methylation and expression probes

plotting

Functions used to plot the results of the analysis. Some are interesting for whole methylome analysis (e.g. plotEWAS) and others for analysis of one genomic region (e.g. plotRDA)

MethylationSet *MethylationSet instances*

Description

Container with the data needed to perform methylation analysis. MethylationSet inherits from eSet and contains meth matrix as assay data member.

Usage

```
methylationSet(betas, phenotypes, annotationDataFrame, annoString = "custom")

## S4 method for signature 'MethylationSet'
betas(object)

## S4 method for signature 'MethylationSet'
getMs(object, threshold = 1e-04)

## S4 method for signature 'MethylationSet'
checkProbes(object)

## S4 method for signature 'MethylationSet'
checkSamples(object)
```

Arguments

betas	Matrix of beta values
phenotypes	Data.frame or AnnotatedDataFrame with the phenotypes
annotationDataFrame	Data.frame or AnnotatedDataFrame with the phenotypes with the annotation of the methylation sites. A column with the chromosomes named chr and a column with the positions names pos are required.
annoString	Character with the name of the annotation used.
object	MethylationSet
threshold	Numeric with the threshold to avoid 0s and 1s.

Details

FeatureData, which contains annotation data, is required to perform any of the analysis.

Value

MethylationSet

Methods (by generic)

- betas: Get beta matrix
- getMs: Get Ms values
- checkProbes: Filter probes with annotation
- checkSamples: Modify a MethylationSet to only contain common samples

Slots

assayData Contains matrices with equal dimensions, and with column number equal to nrow(phenoData). assayData must contain a matrix meth with rows representing features (e.g., methylation probes sets) and columns representing samples.

phenoData See eSet

annotation See eSet

featureData See eSet. fData should contain at least chromosome and positions columns.

Examples

```
showClass("MethylationSet")
```

`multiCorrMethExprs` *Computes the correlation between methylation and expression in a genomic range*

Description

Estimates the correlation between methylation and expression in a range. First, the sets are filtered to only contain the features of the range. Then, a multivariate approach (redundancy analysis) is applied.

Usage

```
multiCorrMethExprs(multiset, vars_meth = NULL, vars_exprs = NULL,
  vars_meth_types = rep(NA, length(vars_meth)), vars_exprs_types = rep(NA,
  length(vars_exprs)), range = NULL)
```

Arguments

`multiset`	`MultiDataSet` containing a `methylation` and an `expression` slots.
`vars_meth`	Character vector with the names of the variables that will be used to obtain the methylation residuals. By default, none is used and residuals are not computed.
`vars_exprs`	Character vector with the names of the variables that will be used to obtain the expression residuals. By default, none is used and residuals are not computed.
`vars_meth_types`	Character vector with the types of the methylation variables. By default, variables type won't be changed.
`vars_exprs_types`	Character vector with the types of the expression variables. By default, variables type won't be changed.
`range`	`GenomicRanges` with the desired range.

Details

When there are known variables that affect methylation and/or expression, their effect can be substracted using a linear model and then the residuals are used.

Value

An `rda` object

MultiDataSet-class *Class* MultiDataSet

Description

The class MultiDataSet is a superior class to store multiple datasets in form of triplets (assayData-phenoData-featureData). The restriction is that the samples of the multiple datasets must be the same.

Usage

```
## S4 method for signature 'MultiDataSet,ExpressionSet'
add.genexp(object, gexpSet,
  warnings = TRUE)

## S4 method for signature 'MultiDataSet,MethylationSet'
add.methy(object, methySet,
  warnings = TRUE)

## S4 method for signature 'MultiDataSet,eSet'
add.set(object, set, dataset.name,
  warnings = TRUE)

## S4 method for signature 'MultiDataSet,SnpSet'
add.snps(object, snpSet, warnings = TRUE)

## S4 method for signature 'MultiDataSet'
names(x)

## S4 method for signature 'MultiDataSet'
sampleNames(object)

## S4 method for signature 'MultiDataSet,ANY,ANY'
x[[i]]

## S4 method for signature 'MultiDataSet,ANY,ANY,ANY'
x[i, j, drop = TRUE]
```

Arguments

object	MultiDataSet
gexpSet	ExpressionSet to be used to fill the slot.
warnings	Logical to indicate if warnings will be displayed.
methySet	MethylationSet to be used to fill the slot.
set	Object derived from eSet to be used to fill the slot.
dataset.name	Character with the name of the slot to be filled.
snpSet	SnpSet to be used to fill the slot.
x	MultiDataSet
i	slot

j	samples
drop	Logical indicating if dropped will be applied.

Details

The names of the three lists (assayData, phenoData and featureData)must be the same.

Value

MultiDataSet

Methods (by generic)

- add.genexp: Method to add a slot of expression to MultiDataSet.
- add.methy: Method to add a slot of methylation to MultiDataSet.
- add.set: Method to add a slot to MultiDataSet.
- add.snps: Method to add a slot of SNPs to MultiDataSet.
- names: Get names of slots
- sampleNames: Get sample names
- [[: Get an eSet from a slot
- [: Subset a MultiDataSet

Slots

assayData List of assayData elements.

phenoData List of AnnotatedDataFrame containing the phenoData of each assayData.

featureData List of AnnotatedDataFrame containing the featureData of each assayData.

return_method List of functions used to create the original eSet objects.

---

normalSNP *Normalize SNPs values*

Description

SNPs values, introduced as numerical, are normalized to be used in lineal models.

Usage

```
normalSNP(snps)
```

Arguments

snps	Numerical vector or matrix representing the SNPs in the form: 0 homozygote recessive, 1 heterozygote, 2 homozygote dominant.

Value

Numerical vector or matrix with the snps normalized.

Examples

```
snps <- c(1, 0, 0, 1, 0, 0, 2, 1, 2)
normSNPs <- normalSNP(snps)
normSNPs
```

---

plotBestFeatures *Plot best n cpgs*

Description

Wrapper of plotCPG that plots the top n features.

Usage

```
plotBestFeatures(set, n = 10, variables = variableNames(set)[1])
```

Arguments

set	AnalysisResults, AnalysisRegionResults, ExpressionSet or MethylationSet
n	Numeric with the number of features to be plotted.
variables	Character vector with the names of the variables to be used in the splitting.

Value

Plots are created on the current graphics device.

See Also

plotFeature

Examples

```
if (require(minfiData)){
 set <- prepareMethylationSet(getBeta(MsetEx)[1:10, ],
 pheno = pData(MsetEx))
 plotBestFeatures(set, 2, variables = "Sample_Group")
 }
```

plotEWAS *Plot a Manhattan plot with the probe results*

Description

Plot log p-value for each chromosome positions. Highlighting cpgs inside a range is allowed.

Usage

```
plotEWAS(object, variable = modelVariables(object)[[1]], range = NULL)
```

Arguments

object	AnalysisResults or AnalysisRegionResults
variable	Character with the variable name used to obtain the probe results. Note: model name should be used. Original variable name might not be valid.
range	GenomicRange whose cpgs will be highlighted

Value

A plot is generated on the current graphics device.

Examples

```
if (require(minfiData)){
betas <- getBeta(MsetEx)[floor(seq(1, nrow(MsetEx), 10000)), ]
set <- prepareMethylationSet(betas, pheno = pData(MsetEx))
methyOneVar <- DAPipeline(set, variable_names = "sex", probe_method = "ls")
plotEWAS(methyOneVar)
}
```

plotFeature *Plot values of a feature*

Description

Plot values of a feature splitted by one or two variables.

Usage

```
plotFeature(set, feat, variables = variableNames(set)[1])
```

Arguments

set	AnalysisResults, AnalysisRegionResults, ExpressionSet or MethylationSet
feat	Numeric with the index of the feature or character with its name.
variables	Character vector with the names of the variables to be used in the splitting. Two variables is the maximum allowed. Note: default values are only valid for MethylationResults objects.

Value

A plot is generated on the current graphics device.

Examples

```
if (require(minfiData)){
 set <- prepareMethylationSet(getBeta(MsetEx)[1:1000, ],
 pheno = pData(MsetEx))
 plotFeature(set, 1, variables = "Sample_Group")
 }
```

`plotQQ` *QQ plot of probe analysis*

Description

Generate a QQ plot using probe results.

Usage

```
plotQQ(object, variable = modelVariables(object)[[1]])
```

Arguments

`object`	`AnalysisResults` or `AnalysisRegionResults`
`variable`	Character with the variable name used to obtain the probe results. Note: model name should be used. Original variable name might not be valid.

Value

A plot is generated on the current graphics device.

Examples

```
if (require(minfiData)){
betas <- getBeta(MsetEx)[floor(seq(1, nrow(MsetEx), 10000)), ]
set <- prepareMethylationSet(betas, pheno = pData(MsetEx))
methyOneVar <- DAPipeline(set, variable_names = "sex", probe_method = "ls")
plotQQ(methyOneVar)
}
```

plotRDA *Plot RDA results*

Description

Plot RDA results

Usage

```
plotRDA(object, n_feat = 5)
```

Arguments

object	AnalysisRegionResults
n_feat	Numeric with the number of cpgs to be highlighted.

Value

A plot is generated on the current graphics device.

Examples

```
if (require(minfiData) & require(GenomicRanges)){
set <- prepareMethylationSet(getBeta(MsetEx), pheno = pData(MsetEx))
range <- GenomicRanges::GRanges(seqnames=Rle("chrY"),
ranges = IRanges(3000000, end=12300000))
rangeNoSNPs <- DARegionAnalysis(set, variable_names = "sex", range = range)
plotRDA(rangeNoSNPs)
}
```

plotRegion *Plot of the region*

Description

Plot of the beta values againts their position. Data is taken from probe analysis. Cpgs with a p-value smaller than 0.05 (without adjusting) are blue and points with a p-value greater than 0.05 are red.

Usage

```
plotRegion(object, variable = modelVariables(object)[[1]], range = NULL)
```

Arguments

object	AnalysisResults or AnalysisRegionResults
variable	Character with the variable name used to obtain the probe results. Note: model name should be used. Original variable name might not be valid.
range	GenomicRange whose cpgs will be shown (only for AnalysisResults objects)

Value

A plot is generated on the current graphics device.

Examples

```
if (require(minfiData) & require(GenomicRanges)){
set <- prepareMethylationSet(getBeta(MsetEx), pheno = pData(MsetEx))
range <- GenomicRanges::GRanges(seqnames=Rle("chrY"),
ranges = IRanges(3000000, end=12300000))
rangeNoSNPs <- DARegionAnalysis(set, variable_names = "sex", range = range)
plotRegion(rangeNoSNPs)
}
```

---

plotRegionR2 *Plot R2 region values*

Description

Plot R2 region values

Usage

```
plotRegionR2(object, feat, ...)
```

Arguments

object	MethylationRegionResults
feat	Numeric with the index of the feature or character with its name.
...	Further arguments passed to plotLM

Value

A plot is generated on the current graphics device.

---

plotVolcano *Make a Volcano plot with the probe results*

Description

Plot log p-value versus the change in expression/methylation.

Usage

```
plotVolcano(object, variable = modelVariables(object)[[1]])
```

Arguments

object	MethylationResults or MethylationRegionResults
variable	Character with the variable name used to obtain the probe results. Note: model name should be used. Original variable name might not be valid.

Value

A plot is generated on the current graphics device.

Examples

```
if (require(minfiData)){
betas <- getBeta(MsetEx)[floor(seq(1, nrow(MsetEx), 10000)), ]
set <- prepareMethylationSet(betas, pheno = pData(MsetEx))
methyOneVar <- DAPipeline(set, variable_names = "sex", probe_method = "ls")
plotEWAS(methyOneVar)
}
```

---

prepareMethylationSet *Generating a* MethylationSet

Description

This function creates a MethylationSet using from a matrix of beta values and a data.frame of phenotypes.

Usage

```
prepareMethylationSet(matrix, phenotypes,
  annotation = "IlluminaHumanMethylation450kanno.ilmn12.hg19",
  chromosome = "chr", position = "pos", genes = "UCSC_RefGene_Name",
  group = "UCSC_RefGene_Group", filterNA_threshold = 0.05,
  verbose = FALSE)
```

Arguments

matrix	Data.frame or a matrix with samples on the columns and cpgs on the rows. A minfi object can be used to.
phenotypes	Data.frame or vector with the phenotypic features of the samples. Samples will be in the rows and variables in the columns. If matrix is a minfi object, phenotypes can be taken from it.
annotation	Character with the name of the annotation package or data.frame or AnnotationDataFrame with the annotation.
chromosome	Character with the column containing chromosome name in the annotation data.
position	chromosome Character with the column containing position coordinate in the annotation data.
genes	Character with the column containing gene names related to the methylation site in the annotation data. (Optional)
group	Character with the column containing the position of the probe related to the gene named in gene column. (Optional)
filterNA_threshold	Numeric with the maximum percentage of NA allowed for each of the probes. If 1, there will be no filtering, if 0 all probes containing at least a NA will be filtered.
verbose	Logical value. If TRUE, it writes out some messages indicating progress. If FALSE nothing should be printed.

Details

prepareMethylationSet is a useful wrapper to create `MethylationSet`. Rigth now, prepareMethylationSet supports two entry points: a `minfi` object and a matrix of betas.

Phenotypes are compulsory and can be supplied as data.frame or AnnotatedDataFrame.

By default, annotation is taken from `minfi` package and `IlluminaHumanMethylation450kanno.ilmn12.hg19` package is used, being the default arguments adapted to use this annotation. To use this annotation, `IlluminaHumanMethylation450kanno.ilmn12.hg19` must be installed and methylation sites must be named like in Illumina 450k chip. Use of this annotation ensures correct results in all the analysis.

If custom annotation is desired, there are two compulsory features: chromosomes and positions. Chromosomes should be supplied in the character form (e.g. chr1). Two additional features will be used during the presentation of results but not during the analyses: genes and group. Genes are the gene names of the genes around the cpg site and group defines the groups of the genes. Both columns will appear in the results but they are not used through the workflow. It should be noticed that `BlockFinder` only supports `minfi` annotation, so it is not advised to be used with custom annotation.

Value

`MethylationSet` with phenotypes and annotation.

Examples

```
if (require(minfiData)){
 betas <- getBeta(MsetEx)[1:1000, ]
 pheno <- pData(MsetEx)
 set <- prepareMethylationSet(betas, pheno)
 }
```

---

preparePhenotype *Process a table of phenotypes*

Description

Given a data.frame containing phenotypic variables, select the desired columns and transform them to the desired types.

Usage

```
preparePhenotype(phenotypes, variable_names, variable_types = rep(NA,
  length(variable_names)))
```

Arguments

`phenotypes`	Data.frame with the phenotypic features
`variable_names`	Vector with the names or the positions of the desired variables.
`variable_types`	Vector with the types of the variables.

Details

preparePhenotype supports five types of variables. Categorical and continuous correspond to factor and numerical types in R. The other three are genomic models as defined in SNPassoc: dominant, recessive and additive. In order to use these types, only two alleles can be present and genotypes should be specified in the form *a/b*.

If transformation of variables is not needed, the variable_types can be passed as a vector of NA.

Value

Data.frame with the columns selected and with the types desired.

Examples

```
pheno <- data.frame(a = sample(letters[1:2], 5, replace = TRUE), b = runif(5),
c = sample(c("a/a","a/b", "b/b"), 5, replace = TRUE))
pheno <- preparePhenotype(pheno, variable_names = c("a", "c"),
variable_types = c("categorical", "dominant"))
pheno
```

RDAset *Calculate RDA for a set*

Description

Perform RDA calculation for a AnalysisRegionResults. Feature values will be considered the matrix X and phenotypes the matrix Y. Adjusting for covariates is done using covariable_names stored in the object.

Usage

```
RDAset(set, equation = NULL)
```

Arguments

set	AnalysisResults
equation	Character with the equation used in the analysis

Value

Object of class rda

See Also

rda

Examples

```
if (require(minfiData)){
set <- prepareMethylationSet(getBeta(MsetEx)[1:50,], pheno = pData(MsetEx))
methyOneVar <- DAPipeline(set, variable_names = "sex", probe_method = "ls")
rda <- RDAset(methyOneVar)
rda
}
```

Index