Package: RAMEN (via r-universe)

September 28, 2026

Title Regional Association of Methylome variability with the Exposome and geNome

Version 2.99.0

Description RAMEN is an R package which goal is to identify Variable Methylated Loci (VML) in microarray DNA methylation data and then, using genome and exposome data, identify which individual genetic (G) or environmental (E) or joint (G+E or GxE) model better explains the VML variability. This package provides estimates of genetic and environmental contribution for each VML, allowing researchers to gain insights into the complex interplay between genetics, environment and DNA methylation.

License GPL (>= 3)

Encoding UTF-8

biocViews DNAMethylation, GenomeWideAssociation, Epigenetics, Regression, SNP, Microarray, MethylationArray, GeneticVariability

Roxygen list(markdown = TRUE)

Suggests BiocStyle, knitr, magick, rmarkdown, ggplot2, tidyr, doParallel, IlluminaHumanMethylation450kanno.ilmn12.hg19, IlluminaHumanMethylationEPICv2anno.20a1.hg38, testthat (>= 3.0.0)

Config/testthat/edition 3

Imports doRNG, dplyr, foreach, GenomicRanges, glmnet, IRanges, matrixStats, methods, relaimpo, S4Vectors, IlluminaHumanMethylationEPICanno.ilm10b4.hg19, stats

VignetteBuilder knitr

Depends R (>= 4.6.0)

LazyData true

URL https://ropensci.github.io/RAMEN/ , https://github.com/ropensci/RAMEN/

BugReports https://github.com/ropensci/RAMEN/issues

Config/roxygen2/version 8.1.0

Config/pak/sysreqs make libbz2-dev libicu-dev liblzma-dev libpng-dev libxml2-dev libssl-dev libx11-dev xz-utils zlib1g-dev

Repository https://ropensci.r-universe.dev

Date/Publication 2026-08-29 00:11:23 UTC

RemoteUrl https://github.com/ropensci/RAMEN

RemoteRef main

RemoteSha 57405c7c8443c0c0039406883902fd8633846da6

Contents

findCisSNPs . . . 2
findVML . . . 4
lmGE . . . 7
medCorVMR . . . 11
nullDistGE . . . 12
selectVariables . . . 15
summarizeVML . . . 18
test_covariates . . . 19
test_environmental_matrix . . . 20
test_genotype_information . . . 21
test_genotype_matrix . . . 21
test_methylation_data . . . 22
ultrastable_cpgs . . . 22

Index **24**

`findCisSNPs` *Find cis SNPs around a set of Variable Methylated Loci (VML)*

Description

Identification of genotyped Single Nucleotide Polymorphisms (SNPs) close to each VML using a distance threshold.

Usage

```
findCisSNPs(VML, genotype_information, distance = 1e+06)
```

Arguments

VML	GRanges object. Must contain a metadata column named "probes", where each element contains a vector with the probes constituting the VML.
genotype_information	A data frame with information about genotyped sites of interest. It must contain the following columns: "CHROM" (chromosome number), "POS" (Genomic basepair position of the SNP (must be an integer), and "ID" (SNP ID). The nomenclature of CHROM must match with the one used in the VML seqnames column (i.e., if VML uses 1, 2, 3, X, Y or Chr1, Chr2, Chr3, ChrX, ChrY, etc. as chromosome number, the genotype_information$CHROM values must be encoded in the same way).
distance	The distance threshold in basepairs to be used to identify cis SNPs. Default is 1 Mb.

Details

For DNAme data, previous studies (e.g. Gibbs, et al., 2010, McClay, et al., 2015) have found that SNPs are more likely to associate with DNAme levels the closer they are to the CpG. We recommend to include SNPs within 500 kb to 1 Mb to capture SNPs with a high potential to associate with DNAme.

Important: please make sure that the positions of the VML data frame and the ones in the genotype information are from the same genome build.

Value

The same VML object with new metadata columns indicating the cis SNPs identified for each VML and the number of SNPs surrounding each VML in the specified window

Examples

```
## Find VML in test data
VML <- findVML(
  methylation_data = RAMEN::test_methylation_data,
  array_manifest = "IlluminaHumanMethylationEPICv1",
  cor_threshold = 0,
  var_method = "variance",
  var_distribution = "ultrastable",
  var_threshold_percentile = 0.99,
  max_distance = 1000
)
## Find cis SNPs around VML
# Use only 5 for demonstration purposes
VML_with_cis_snps <- findCisSNPs(
  VML = VML$VML[1:5, ],
  genotype_information = RAMEN::test_genotype_information,
  distance = 1e6
)
```

`findVML` *Identify Variable Methylated Loci in microarrays*

Description

Identifies Highly Variable Probes (HVP) and groups them into Variable Methylated Loci (VML) given an Illumina manifest.The output of this function provides the HVPs, and the identified VML, which are made of Variable Methylated Regions and sparse Variable Methylated Probes. See Details below for more information.

Usage

```
findVML(
  methylation_data,
  array_manifest,
  cor_threshold = 0.15,
  var_method = "variance",
  var_distribution = "ultrastable",
  var_threshold_percentile = 0.99,
  max_distance = 1000
)
```

Arguments

`methylation_data`
: A data frame containing M or B values, with samples as columns and probes as rows. Data is expected to have already passed through quality control and cleaning steps. Rows must be the CpG probe IDs.

`array_manifest`
: Information about the probes on the array in a format compatible with the Bioconductor annotation packages. The user can specify one of the supported human microarrays ("IlluminaHumanMethylation450k" with the hg19 genome build, "IlluminaHumanMethylationEPICv1" with the hg19 genome build, or "IlluminaHumanMethylationEPICv2" with the hg38 genome build), or provide a manifest. The manifest requires the probe names as row names, and the following columns: "chr" (chromosome); "pos" (genomic location of the probe in the genome); and "strand" (this is very important to set up, since the VMRs will only be created based on CpGs on the same strand; if the positions are reported based on a single DNA strand, this should contain either a vector of only "+", "-" or "*" for all of the probes).

`cor_threshold`
: Numeric value (0-1) to be used as the median pearson correlation threshold for identifying VMRs (i.e. all VMRs will have a median pairwise probe correlation higher than this threshold).

`var_method`
: A string indicating the metric to use to represent variability in the data set. The options are "mad" (median absolute deviation) or "variance".

`var_distribution`
: A string indicating which probes in the data set should be used to create a variability distribution; the threshold to identify Highly Variable Probes (determined also with the var_threshold_percentile argument) is established based on this distribution. The options 1 is "ultrastable" (a subset of CpGs that are stably methylated/unmethylated across human tissues and developmental states described by Edgar R., et al. in 2014). This option is recommended, especially if you want to compare different populations or tissues, as the threshold value should be comparable. On the other hand, the user can use option 2: "all" (all probes in the data set). The "ultrastable" option is only compatible with Illumina human microarrays. The default is "ultrastable".

`var_threshold_percentile`
: The percentile (0-1) to be used as cutoff to define Highly Variable Probes (which are then grouped into VML). If using the variability of the "ultrastable" probes, we recommend a high threshold (default is 0.99), since these probes are expected to display a very low variation in human tissues. If using the variability of "all" probes, we recommend using a percentile of 0.9 since it captures the top 10% most variable probes, which has been traditionally used in studies. It is important to note that the top 10% most variable probes will capture the same amount of probes in a data set regardless of their overall variability levels, which might differ between tissues or populations.

`max_distance`
: Maximum distance in base pairs allowed for two probes to be grouped into a region. The default is 1000.

Details

This function identifies HVPs based on variance or MAD scores, and groups them into VML, which are defined as genomic locations with high DNA methylation variability.To best capture methylome variability patterns in microarrays, we identify two types of VML: Variably Methylated Regions (VMRs) and sparse Variably Methylated Probes (sVMPs) .

In one hand, we defined VMRs as two or more proximal highly variable probes (default: < 1kb apart) with correlated DNAme level (default: r > 0.15). Modelling DNAme variability through regions rather than individual CpGs provides several methodological advantages in association studies, since CpGs display a significant correlation for co-methylation when they are close (less than or equal to 1 kilobase). Modelling DNAme variability through regions rather than individual CpGs provides several methodological advantages in association studies, since CpGs display a significant correlation for co-methylation when they are close (less than or equal to 1 kilobase). Modelling DNAme variability through regions rather than individual CpGs provides several methodological advantages in association studies, since CpGs display a significant correlation for co-methylation when they are close (less than 1 kilobase). Some of these advantages include increasing statistical power by testing redundant probes only once, reducing false-positives driven by one problematic probe in a region, and improving comparability between studies that analyze the same genomic region but measure distinct CpGs due to microarray design differences. In contrast with Differentially Methylated Regions (DMRs), a different class of regional construct used in the field, VMRs denote regions with high inter-individual variability in methylation levels within a single population, while DMRs represent regions where DNA methylation differs significantly across a variable of interest.

In addition to traditional VMRs, we also identified sparse Variably Methylated Probes (sVMPs), a second type of VML that takes into account the sparse and non-uniformly distributed coverage

of CpGs in microarrays to tailor our analysis to this DNAme platform. sVMPs aimed to retain genomic regions with high DNAme variability measured by single probes, where probe grouping based on proximity and correlation is therefore not applicable. This is particularly relevant in the Illumina EPIC v1 array, where most covered regulatory regions (up to 93%) are represented by just one probe. Notably, based on empirical comparisons with whole-genome bisulfite sequencing data, these single probes are mostly representative of local regional DNAme levels due to their positioning (98.5-99.5%)

This function uses GenomicRanges::reduce() to group the regions, which is strand-sensitive. In the Illumina microarrays, the MAPINFO for all the probes is usually provided for the + strand. If you are using this array, we recommend to first convert the strand of all the probes to "+".

This function supports parallel computing for increased speed. To do so, you have to set the parallel backend in your R session BEFORE running the function (e.g., *doParallel::registerDoParallel(4)*). After that, the function can be run as usual. When working with big datasets, the parallel backend might throw an error if you exceed the maximum allowed size of globals exported for future expression. This can be fixed by increasing the allowed size (e.g. running *options(future.globals.maxSize= +Inf)*)

Note: this function does not exclude sex chromosomes. If you want to exclude them, you can do so in the methylation_data object before running the function.

Value

A list with the following elements:

- $var_score_threshold: threshold used to define Highly Variable Probes (mad or variance, depending on the specified choice).
- $highly_variable_probes: a data frame with the probes that passed the variability score threshold imposed by the user, and their variability score (MAD score or variance).
- $VML: a GRanges object listing VMRs (regions composed of two or more contiguous, correlated and proximal Highly Variable Probes), and sVMPs (highly variable probes without neighboring CpGs measured in *max_distance* on the array).

Examples

```
# Evaluate sequentially
foreach::registerDoSEQ()
VML <- RAMEN::findVML(
  methylation_data = RAMEN::test_methylation_data,
  array_manifest = "IlluminaHumanMethylationEPICv1",
  cor_threshold = 0.15,
  var_method = "variance",
  var_distribution = "ultrastable",
  var_threshold_percentile = 0.99,
  max_distance = 1000
)
```

`lmGE` *Fit linear G, E, G+E and GxE models and select the winning model*

Description

For a set of Variable Methylated Loci (VML), this function fits a set of genotype (G), environment (E), pairwise additive (G + E) or pairwise interaction (G x E) models, one variable at a time, and selects the best fitting one. Additional information for each winning model is provided, such as its R2, its R2 increase comparing it to a basal model (i.e., a model only fitted with the concomitant variables), the delta AIC/BIC to the next best model from a different category, and the explained variance decomposed for the G, E and GxE components (when applicable). If a VML has no variables selected in the selected_variables object, it will be returned with "B" (basal) as the best model (interpreted as no G or E associated effect). For guidance on interpretation, please build and read the package's vignette.

Usage

```
lmGE(
  selected_variables,
  summarized_methyl_VML,
  genotype_matrix,
  environmental_matrix,
  covariates = NULL,
  model_selection = "AIC"
)
```

Arguments

`selected_variables`
: A data frame obtained with *RAMEN::selectVariables()*. This data frame must contain three columns: 'VML_index' with characters of an unique ID of each VML; ´selected_genot' and 'selected_env' with the SNPs and environmental variables, respectively, that will be used for fitting the genotype (G), environment (E), additive (G + E) or interaction (G x E) models. The columns 'selected_env' and 'selected_genot' must contain lists as elements; VML with no environmental or genotype selected variables must contain an empty list (i.e., list(NULL), list(NA), list("") or list(character(0))).

`summarized_methyl_VML`
: A matrix containing each individual's VML summarized methylation. It is suggested to use the output of *RAMEN::summarizeVML()*.Rows must reflects individuals, and columns VML The names of the columns must correspond to the index of said VML, and it must match the index of VML_wSNPs$VML_index. The names of the rows must correspond to the sample IDs, and must match with the IDs of the other matrices.

`genotype_matrix`
: A matrix of number-encoded genotypes. Columns must correspond to samples, and rows to SNPs. We suggest using a gene-dosage model, which would encode

the SNPs ordinally depending on the genotype allele charge, such as 2 (AA), 1 (AB) and 0 (BB). The column names must correspond with individual IDs.

`environmental_matrix`
: A matrix of environmental variables. Only numeric values are supported. In case of factor variables, it is recommended to encode them as numbers or recode them into dummy variables if there are more than two levels. Columns must correspond to environmental variables and rows to individuals. Row names must be the individual IDs.

`covariates`
: A matrix containing the covariates (i.e., concomitant variables / variables that are not the ones you are interested in) that will be adjusted for in the final GxE models (e.g., cell type proportions, age, etc.). Each column should correspond to a covariate and each row to an individual. Row names must correspond to the individual IDs.

`model_selection`
: Which metric to use to select the best model for each VML. Supported options are "AIC" or BIC".

Details

This function supports parallel computing for increased speed. To do so, you have to set the parallel backend in your R session before running the function (e.g., *doParallel::registerDoParallel(4)*). After that, the function can be run as usual. It is recommended to also set options(future.globals.maxSize= +Inf).

For each VML, this function computes a set of models using the variables indicated in the selected_variables object. From the indicated G and E variables, lmGE() fits four groups of models:

- G: Genetics model - fitted one SNP at a time.
- E: Environmental model - fitted one environmental variable at a time.
- G+E: Additive model - fitted for each pairwise combination of G and E variables indicated in selected_variables.
- GxE: Interaction model - fitted for each pairwise combination of G and E variables indicated in selected_variables.

These models are fit only if the VML has G or E variables in the selected_variables object. If a VML does not have neither G nor E variables, that VML will be ignored and will be returned in the output object with "B" (baseline) as the best explanatory model.

Model selection

Following the model fitting stage, the best model **per group** is selected using Akaike Information Criterion (AIC) or Bayesian Information Criterion (BIC). Both of these metrics are statistical approaches to select the best model in the same data set, and they have strengths and limitations that make them excel in different situations. We recommend using AIC because BIC assumes that the true model is in the set of compared models. Since this function fits models with individual variables, and we assume that DNAme variability is more likely to be influenced by more than one single SNP/environmental exposure at a time, we hypothesize that in most cases, the true model will not be in the set of compared models. Also, AIC excels in situations where all models in the model space are "incomplete", and AIC is preferentially used in cases where the true underlying function is unknown and our selected model could belong to a very large class of functions where

the relationship could be pretty complex. It is worth mentioning however that, both metrics tend to pick the same model in a large number of scenarios. We suggest the users to read Arijit Chakrabarti & Jayanta K. Ghosh, 2011 for further information about the difference between these metrics.

After selecting the best model per group (G,E,G+E pr GxE), the model with the lowest AIC or BIC is declared as the winning model. The delta AIC/BIC and difference of R2 is computed relative to the model with the second lowest AIC/BIC (i.e., the best model from a different group to the winning one), and reported in the final object.

Analysis of variance and variance decomposition

Finally, the variance is decomposed and the relative R2 contribution of each of the variables of interest (G, E and GxE) is reported. This decomposition is done using the relaimpo R package, using the Lindeman, Merenda and Gold (lmg) method, which is based on the heuristic approach of averaging the relative R contribution of each variable over all input orders in the linear model. The estimation of the partitioned R2 of each factor in the models was conducted keeping the covariates always in the model as first entry (i.e., the variables specified in covariates did not change order). For further information, we suggest the users to read the documentation and publication of the relaimpo R package (Grömping, 2006).

Value

A data frame with the following columns:

- VML_index: The unique ID of the VML
- model_group: The group to which the winning model belongs to (i.e., G, E, G+E or GxE)
- variables: The variable(s) that are present in the winning model (excluding the covariates, which are included in all the models)
- tot_r_squared: R squared of the winning model
- g_r_squared: Estimated R2 allocated to the G in the winning model, if applicable.
- e_r_squared: Estimated R2 allocated to the E in the winning model, if applicable.
- gxe_r_squared: Estimated R2 allocated to the interaction in the winning model (GxE), if applicable.
- AIC/BIC: AIC or BIC metric from the best model in each VML (depending on the option specified in the argument model_selection).
- second_winner: The second group that possesses the next best model after the winning one (i.e., G, E, G+E or GxE). This column may have NA if the variables in selected_variables correspond only to one group (G or E), so that there is no other model groups to compare to.
- delta_aic/delta_bic: The difference of AIC or BIC value (depending on the option specified in the argument model_selection) of the winning model and the best model from the second_winner group (i.e., G, E, G+E or GxE). This column may have NA if the variables in selected_variables correspond only to one group (G or E), so that there is no other groups to compare to.
- delta_r_squared: The R2 of the winning model - R2 of the second winner model. This column may have NA if the variables in selected_variables correspond only to one group (G or E), so that there is no other groups to compare to.
- basal_AIC/basal_BIC: AIC or BIC of the basal model (i.e., model fitted only with the concomitant variables specified in the *covariates* argument)

- basal_rsquared: The R2 of the basal model (i.e., model fitted only with the concomitant variables specified in the *covariates* argument)

Examples

```
# Evaluate sequentially
foreach::registerDoSEQ()
## Find VML in test data
VML <- RAMEN::findVML(
  methylation_data = RAMEN::test_methylation_data,
  array_manifest = "IlluminaHumanMethylationEPICv1",
  cor_threshold = 0,
  var_method = "variance",
  var_distribution = "ultrastable",
  var_threshold_percentile = 0.99,
  max_distance = 1000
)
## Find cis SNPs around VML
VML_with_cis_snps <- RAMEN::findCisSNPs(
  # Use only 5 for demonstration purposes
  VML = VML$VML [1:5, ],
  genotype_information = RAMEN::test_genotype_information,
  distance = 1e6
)

## Summarize methylation levels in VML
summarized_methyl_VML <- RAMEN::summarizeVML(
  methylation_data = RAMEN::test_methylation_data,
  VML = VML_with_cis_snps
)

## Select relevant genotype and environmental variables
selected_vars <- RAMEN::selectVariables(
  VML_wSNPs = VML_with_cis_snps,
  genotype_matrix = RAMEN::test_genotype_matrix,
  environmental_matrix = RAMEN::test_environmental_matrix,
  covariates = RAMEN::test_covariates,
  summarized_methyl_VML = summarized_methyl_VML,
  seed = 1
)

## Fit G, E, G+E and GxE models and select the winning one
lmge_res <- RAMEN::lmGE(
  selected_variables = selected_vars,
  summarized_methyl_VML = summarized_methyl_VML,
  genotype_matrix = RAMEN::test_genotype_matrix,
  environmental_matrix = RAMEN::test_environmental_matrix,
  covariates = RAMEN::test_covariates,
  model_selection = "AIC"
)
```

`medCorVMR` *Compute the median probe methylation pearson correlation for each Variable Methylated Region (VMR).*

Description

This function will take a GRanges object converted into a data frame, where each row corresponds to a Variable Methylated Region. Then, it computes the pairwise correlation of the probes of each VMR and reports its median pairwise probe correlation.

Usage

```
medCorVMR(VML, methylation_data)
```

Arguments

`VML`	GRanges object. Must contain a metadata column named "probes", where each element contains a vector with the probes constituting the VML.
`methylation_data`	A data frame containing M or B values, with samples as columns and probes as rows. Data is expected to have already passed through quality control and cleaning steps. Rows must be the CpG probe IDs.

Details

This function supports parallel computing for increased speed. To do so, you have to set the parallel backend in your R session before running the function (e.g., *doParallel::registerDoParallel(4)*). After that, the function can be run as usual. It is recommended to also set options(future.globals.maxSize= +Inf).

Value

A GRanges object like VML with an extra column per region containing the median pairwise correlation.

Examples

```
# Evaluate sequentially
foreach::registerDoSEQ()
# Create a VML object
VML <- GenomicRanges::GRanges(seqnames = c("chr21", "chr21"),
      ranges = IRanges::IRanges(start = c(10861376, 10862171),
                      end = c(10862507, 10883548)),
      probes = I(list(
        c("cg15043638", "cg18287590", "cg17975851"),
        c("cg13893907", "cg17035109", "cg06187584")))
      )
```

```
# Compute median correlation for each VMR
medCorVMR(VML = VML, methylation_data = RAMEN::test_methylation_data)
```

`nullDistGE` *Simulate a delta R squared null distribution of G and E effects on DNAme variability*

Description

This function simulates the delta R squared distribution under the null hypothesis of G and E having no association with DNA methylation (DNAme) variability through a permutation analysis. To do so, this function shuffles the G and E variables in the dataset, which is followed by a the variable selection and modelling steps with *selectVariables()* and *lmGE()*.These steps are repeated several times as indicated in the *permutations* parameter. By using shuffled G and E data, we simulate the increase of R2 that would be observed in random data using the RAMEN methodology.

Usage

```
nullDistGE(
  VML_wSNPs,
  genotype_matrix,
  environmental_matrix,
  summarized_methyl_VML,
  permutations = 5,
  covariates = NULL,
  seed = NULL,
  model_selection = "AIC"
)
```

Arguments

`VML_wSNPs`	GRanges object produced by *RAMEN::findCisSNPs()*. Must contain the following metadata columns: "VML_index" (a unique ID for each VML in VML_df AS CHARACTERS) and "SNP" (a column with a list as observation, containing the name of the SNPs surrounding the corresponding VML). The SNPs contained in the "SNP" column must be present in the object that is indicated in the genotype_matrix argument. VML_wSNPs must contain all the VML contained in summarized_methyl_VML. VML with no surrounding SNPs must have an empty list in the SNP column (either list(NULL), list(NA), list("") or list(character(0))).
`genotype_matrix`	A matrix of number-encoded genotypes. Columns must correspond to samples, and rows to SNPs. We suggest using a gene-dosage model, which would encode the SNPs ordinally depending on the genotype allele charge, such as 2 (AA), 1 (AB) and 0 (BB). The column names must correspond with individual IDs.

`environmental_matrix`
: A matrix of environmental variables. Only numeric values are supported. In case of factor variables, it is recommended to encode them as numbers or recode them into dummy variables if there are more than two levels. Columns must correspond to environmental variables and rows to individuals. Row names must be the individual IDs.

`summarized_methyl_VML`
: A matrix containing each individual's VML summarized methylation. It is suggested to use the output of *RAMEN::summarizeVML()*.Rows must reflects individuals, and columns VML The names of the columns must correspond to the index of said VML, and it must match the index of VML_wSNPs$VML_index. The names of the rows must correspond to the sample IDs, and must match with the IDs of the other matrices.

`permutations`
: Numer of permutation analyses to run.

`covariates`
: A matrix containing the covariates (i.e., concomitant variables / variables that are not the ones you are interested in) that will be adjusted for in the final GxE models (e.g., cell type proportions, age, etc.). Each column should correspond to a covariate and each row to an individual. Row names must correspond to the individual IDs.

`seed`
: An integer number that initializes a pseudo-random number generator. Random numbers in this function are created during the lambda cross validation and the LASSO stages. Setting a seed is highly encouraged for result reproducibility. **The seed is applied for the duration of this call only; the global random stream is restored when the function returns**.

`model_selection`
: Which metric to use to select the best model for each VML. Supported options are "AIC" or BIC".

Details

The core pipeline from the RAMEN package identifies the best explanatory model per VML. However, despite these models being winners in comparison to models including any other G/E variable(s) in the dataset, some winning models might perform no better than what we would expect by chance. Therefore, the goal of this function is to create a distribution of increase in R2 under the null hypothesis of G and E having no associations with DNAme. The null distribution is obtained through shuffling the G and E variables in a given dataset and conducting the variable selection and G/E model selection. That way, we can simulate how much additional variance would be explained by the models defined as winners by the RAMEN methodology in a scenario where the G and E associations with DNAme are randomized. This distribution can be then used to filter out winning models in the non-shuffled dataset that do not add more to the explained variance of the basal model than what randomized data do.

Under the assumption that after adjusting for the concomitant variables all VML across the genome follow the same behavior regarding an increment of explained variance with randomized G and E data, we can pool the delta R squared values from all VML to create a null distribution taking advantage of the high number of VML in the dataset. This assumption decreases significantly the number of permutations required to create a null distribution and reduces the computational time. For further information please read the RAMEN paper (https://doi.org/10.1186/s13059-025-03864-4).

Reproducibility and the use of the seed

Random numbers are drawn in this function when the permutation orders are created, and again inside each permutation during the cross-validation and LASSO stages of *selectVariables()*. Setting *seed* makes the whole run reproducible.

Note that the same *seed* value is handed to *selectVariables()* in every permutation. For a given VML the cross-validation folds are therefore identical from one permutation to the next, and what differs between permutations is the shuffled G and E data. The permutations are, in that sense, not fully independent draws with respect to the cross-validation randomness. This holds one source of variability fixed rather than biasing the delta R squared values, and its practical effect is small because the null distribution pools delta R squared across all the VML in the dataset, which is where nearly all of its draws come from. It is nonetheless worth keeping in mind when interpreting the spread of the distribution.

The seed is applied for the duration of this call only: the random number generator state found on entry is restored when the function returns, so a seeded run leaves the global random stream untouched.

Value

A data frame with the following columns:

- VML_index: The unique ID of the VML.
- model_group: The group to which the winning model belongs to (i.e., G, E, G+E or GxE)
- tot_r_squared: R squared of the winning model
- R2_difference: the increase in R squared obtained by including the G/E variable(s) from the winning model (i.e., the R squared difference between the winning model and the model only with the concomitant variables specified in *covariates*; tot_r_squared - basal_rsquared in the lmGE output)
- AIC_difference/BIC_difference: the AIC/BIC difference between the winning model and the model only with the concomitant variables specified in *covariates*; BIC/AIC - basal_BIC/basal_BIC in the lmGE output)

Examples

```
# Evaluate sequentially
foreach::registerDoSEQ()
## Find VML in test data
VML <- findVML(
  methylation_data = test_methylation_data,
  array_manifest = "IlluminaHumanMethylationEPICv1",
  cor_threshold = 0,
  var_method = "variance",
  var_distribution = "ultrastable",
  var_threshold_percentile = 0.99,
  max_distance = 1000
)
## Find cis SNPs around VML
VML_with_cis_snps <- findCisSNPs(
  # Use only 5 for demonstration purposes
```

```
  VML = VML$VML[1:5, ],
  genotype_information = test_genotype_information,
  distance = 1e6
)

## Summarize methylation levels in VML
summarized_methyl_VML <- summarizeVML(
  methylation_data = test_methylation_data,
  VML = VML_with_cis_snps
)

## Simulate null distribution of G and E contributions on DNAme variability
## We will only run one permutation for demonstration purposes
null_dist <- nullDistGE(
  VML_wSNPs = VML_with_cis_snps,
  genotype_matrix = test_genotype_matrix,
  environmental_matrix = test_environmental_matrix,
  summarized_methyl_VML = summarized_methyl_VML,
  # Use one permutation for demonstration purposes
  permutations = 1,
  covariates = test_covariates,
  seed = 1,
  model_selection = "AIC"
)
```

`selectVariables` *Selection of relevant environment and genotype variables associated with Variably Methylated Loci (VML)*

Description

For each VML, this function selects potentially relevant genotype and environmental variables associated with DNA methylation levels of said VML using LASSO. See details below for more information.

Usage

```
selectVariables(
  VML_wSNPs,
  genotype_matrix,
  environmental_matrix,
  covariates = NULL,
  summarized_methyl_VML,
  seed = NULL
)
```

Arguments

VML_wSNPs	GRanges object produced by *RAMEN::findCisSNPs()*. Must contain the following metadata columns: "VML_index" (a unique ID for each VML in VML_df AS CHARACTERS) and "SNP" (a column with a list as observation, containing the name of the SNPs surrounding the corresponding VML). The SNPs contained in the "SNP" column must be present in the object that is indicated in the genotype_matrix argument. VML_wSNPs must contain all the VML contained in summarized_methyl_VML. VML with no surrounding SNPs must have an empty list in the SNP column (either list(NULL), list(NA), list("") or list(character(0))).
genotype_matrix	A matrix of number-encoded genotypes. Columns must correspond to samples, and rows to SNPs. We suggest using a gene-dosage model, which would encode the SNPs ordinally depending on the genotype allele charge, such as 2 (AA), 1 (AB) and 0 (BB). The column names must correspond with individual IDs.
environmental_matrix	A matrix of environmental variables. Only numeric values are supported. In case of factor variables, it is recommended to encode them as numbers or recode them into dummy variables if there are more than two levels. Columns must correspond to environmental variables and rows to individuals. Row names must be the individual IDs.
covariates	A matrix containing the covariates (i.e., concomitant variables / variables that are not the ones you are interested in) that will be adjusted for in the final GxE models (e.g., cell type proportions, age, etc.). Each column should correspond to a covariate and each row to an individual. Row names must correspond to the individual IDs.
summarized_methyl_VML	A matrix containing each individual's VML summarized methylation. It is suggested to use the output of *RAMEN::summarizeVML()*.Rows must reflects individuals, and columns VML The names of the columns must correspond to the index of said VML, and it must match the index of VML_wSNPs$VML_index. The names of the rows must correspond to the sample IDs, and must match with the IDs of the other matrices.
seed	An integer number that initializes a pseudo-random number generator. Random numbers in this function are created during the lambda cross validation and the LASSO stages. Setting a seed is highly encouraged for result reproducibility. **The seed is applied for the duration of this call only; the global random stream is restored when the function returns**.

Details

selectVariables() uses LASSO, which is an embedded variable selection method that penalizes models that are more complex (i.e., that contain more variables) in favor of simpler models (i.e. that contain less variables), but not at the expense of reducing predictive power. Using LASSO's variable screening property (with high probability, the LASSO estimated model includes the substantial covariates and drops the redundant ones) this function selects genotype and environment variables with potential relevance in the Variable Methylated Loci (VML) dataset (see also Bühlmann and van

de Geer, 2011). For each VML, LASSO is run three times: 1) including only the genotype variables for the selection step, 2) including only the environmental variables for the selection step, and 3) Including both the genotype and environmental variables in the selection step. This is done to ensure that the function captures the variables that are relevant within their own category (e.g., SNPs that are strongly associated with the DNAme levels of a VML in the presence of the rest of the SNPs) or in the presence of the variables of the other category (e.g. SNPs that are strongly associated with the DNAme levels of a VML in the presence of the rest of BOTH the SNPs AND environmental variables). Every time LASSO is run, the basal covariates (i.e., concomitant variables)indicated in the argument *covariates* are not penalized (i.e., those variables are always included in the models and their coefficients are not subjected to shrinkage). That way, only the most promising E and G variables in the presence of the concomitant variables will be selected.

Each LASSO model uses a tuned lambda that minimizes the 5-fold cross-validation error within its corresponding data. This function uses the lambda.min value in contrast to lambda.1se because its goal within the RAMEN package is to use LASSO to reduce the number of variables that are going to be used next for fitting pairwise interaction models in *lmGE()*. Since at this step variables are being selected based only on main effects, it is preferable to cast a "wider net" and select a slightly higher number of variables that could potentially have a strong interaction effect when paired with another variable. Furthermore, since in this case LASSO is being used as a screening procedure to select variables that will be fit separately in independent models and compared, the overfitting issue of using lambda.min does not impose a big concern. After finding the best lambda value, the sequence of models is fit by coordinate descent using *glmnet()*. Random numbers in this function are created during the lambda cross validation and the LASSO stages. Setting a seed is highly encouraged for result reproducibility using the *seed* argument. The seed is applied for the duration of this call only: the random number generator state found on entry is restored when the function returns, so a seeded run leaves the global random stream untouched.

This function supports parallel computing for increased speed. To do so, you have to set the parallel back-end in your R session before running the function (e.g., *doParallel::registerDoParallel(4)*). After that, the function can be run as usual. It is recommended to also set options(future.globals.maxSize= +Inf). Please make sure that your data has no NAs and it's all numerical, since the LASSO implementation we use does not support missing or non-numerical values.

Note: If you want to conduct the variable selection step only on the genotype, you can set *environmental_matrix = NULL*. In that case only the genotype model is fitted, and the *selected_env* column of the output is empty for every VML. A *genotype_matrix* is always required.

Value

A data frame with three columns:

- VML_index: Unique VML ID.
- selected_genot: Column containing lists as values with the selected SNPs.
- selected_env: Column containing lists as values with the selected environmental variables.

Examples

```
## Find VML in test data
# Evaluate sequentially
foreach::registerDoSEQ()
VML <- RAMEN::findVML(
```

```
  methylation_data = RAMEN::test_methylation_data,
  array_manifest = "IlluminaHumanMethylationEPICv1",
  cor_threshold = 0,
  var_method = "variance",
  var_distribution = "ultrastable",
  var_threshold_percentile = 0.99,
  max_distance = 1000
)
## Find cis SNPs around VML
VML_with_cis_snps <- RAMEN::findCisSNPs(
  # Use only 5 for demonstration purposes
  VML = VML$VML[1:5, ],
  genotype_information = RAMEN::test_genotype_information,
  distance = 1e6
)

## Summarize methylation levels in VML
summarized_methyl_VML <- RAMEN::summarizeVML(
  methylation_data = RAMEN::test_methylation_data,
  VML = VML_with_cis_snps
)

## Select relevant genotype and environmental variables
selected_vars <- RAMEN::selectVariables(
  VML_wSNPs = VML_with_cis_snps,
  genotype_matrix = RAMEN::test_genotype_matrix,
  environmental_matrix = RAMEN::test_environmental_matrix,
  covariates = RAMEN::test_covariates,
  summarized_methyl_VML = summarized_methyl_VML,
  seed = 1
)
```

`summarizeVML` *Summarize the methylation states of Variable Methylated Loci (VML)*

Description

This function computes a representative methylation score for each Variable Methylated Locus (VML) in a dataset. It returns a data frame with the median methylation of each region per individual. For each VML in a dataset, returns a with the median methylation of that region (columns) per individual (rows) as representative score.

Usage

```
summarizeVML(VML, methylation_data)
```

Arguments

VML	GRanges object. Must contain a metadata column named "probes", where each element contains a vector with the probes constituting the VML.
methylation_data	A data frame containing M or B values, with samples as columns and probes as rows. Data is expected to have already passed through quality control and cleaning steps. Rows must be the CpG probe IDs.

Details

This function supports parallel computing for increased speed. To do so, you have to set the parallel backend in your R session BEFORE running the function (e.g., *doParallel::registerDoParallel(4)*). After that, the function can be run as usual.

Value

A matrix with samples as rows, and VML as columns. The value inside each cell corresponds to the summarized methylation value of said VML in the corresponding individual. The column names correspond to the VML_index.

Examples

```
## Find VML in test data
# Evaluate sequentially
foreach::registerDoSEQ()
VML <- findVML(
  methylation_data = test_methylation_data,
  array_manifest = "IlluminaHumanMethylationEPICv1",
  cor_threshold = 0,
  var_method = "variance",
  var_distribution = "ultrastable",
  var_threshold_percentile = 0.99,
  max_distance = 1000
)

## Summarize methylation states of the found VML
summarized_VML <- summarizeVML(
  # Use only 5 for demonstration purposes
  VML = VML$VML[1:5, ],
  methylation_data = test_methylation_data
)
```

---

test_covariates *Concomitant variable example data set*

Description

Concomitant variable drawn from a normal distribution

Usage

```
test_covariates
```

Format

`test_covariates`**:**

A data frame with 30 rows and 1 column:

rownames Individual IDs

covar1 Concomitant variable drawn from a normal distribution with mean = 0 and sd = 1

Examples

```
summary(test_covariates)
```

---

test_environmental_matrix

Environmental exposures example data set

Description

One hundred environmental exposure variables drawn from a normal distribution with mean=0 and sd=1 for 30 individuals.

Usage

```
test_environmental_matrix
```

Format

`test_environmental_matrix`**:**

A data frame with 30 rows and 100 columns:

rownames Individual ID

E1:100 100 example environmental exposures; column names correspond to exposure IDs

Examples

```
summary(test_environmental_matrix)
```

test_genotype_information *Genotype metadata example data set*

Description

Genotype chromosome 21 position metadata obtained from a private genotyping data set.

Usage

```
test_genotype_information
```

Format

`test_genotype_information`**:**

A data frame with 8539 rows and 3 columns:

CHROM Chromosome

POS Probe genomic position (h19)

ID SNP ID

Examples

```
summary(test_genotype_information)
```

test_genotype_matrix *Genotype matrix example*

Description

Genotype matrix example using a gene-dosage model, which encodes the SNPs ordinally depending on the genotype allele charge, such as 2 (AA), 1 (AB) and 0 (BB). Valus were drawn from a binomial distribution with size 2 and probability 0.5.

Usage

```
test_genotype_matrix
```

Format

`test_genotype_matrix`**:**

A data frame with 8,539 rows and 30 columns:

rownames SNP ID

ID1:30 Individual's 1 to 30 genotypes; column names correspond to individual IDs ...

Examples

```
summary(test_genotype_matrix)
```

`test_methylation_data` *Methylation data matrix example*

Description

Simulated M values of the 3000 probes selected in *test_array_manifest* for 30 individuals. Values were converted from beta values, which were drawn from a bimodal Beta distribution.

Usage

```
test_methylation_data
```

Format

`test_methylation_data`**:**

A data frame with 3,000 rows and 30 columns:

rownames Probe IDs (column TargetID in the EPIC array)

ID1:30 DNAme profile of individuals 1 to 30; column names correspond to individual IDs ...

Examples

```
summary(test_methylation_data)
```

`ultrastable_cpgs` *Ultrastable probes*

Description

This data set contains the list of ultrastable probes identified by Rachel Edgar et. al.,(2014). This publication identified ultrastable CpGs across many tissues and conditions using the Illumina 450k array. Ultrastable probes are defined as CpGs consistently methylated or unmethylated in every sample (1,737 samples from 30 publically available studies). These CpGs are used to create a "null DNAme variance" distribution in the RAMEN package, from which a threshold is taken to identify Highly Variable Probes.

Usage

```
ultrastable_cpgs
```

Format

`ultrastable_cpgs`**:**

A vector with the name of the 15,224 ultrastable probes identified by Edgar et al. (2014). The name of the probes are based on the Illumina 450k manifest.

Source

https://static-content.springer.com/esm/art%3A10.1186%2F1756-8935-7-28/MediaObjects/13072_2014_333_MOESM2_E

Examples

```
head(ultrastable_cpgs)
```

Index

∗ **datasets**
    test_covariates, 19
    test_environmental_matrix, 20
    test_genotype_information, 21
    test_genotype_matrix, 21
    test_methylation_data, 22
    ultrastable_cpgs, 22

findCisSNPs, 2
findVML, 4

lmGE, 7

medCorVMR, 11

nullDistGE, 12

selectVariables, 15
summarizeVML, 18

test_covariates, 19
test_environmental_matrix, 20
test_genotype_information, 21
test_genotype_matrix, 21
test_methylation_data, 22

ultrastable_cpgs, 22