Package 'REMP'

November 12, 2026

Type Package

Title Repetitive Element Methylation Prediction

Version 1.36.0

Description Machine learning-based tools to predict DNA methylation of locus-specific repetitive elements (RE) by learning surrounding genetic and epigenetic information. These tools provide genomewide and single-base resolution of DNA methylation prediction on RE that are difficult to measure using array-based or sequencing-based platforms, which enables epigenome-wide association study (EWAS) and differentially methylated region (DMR) analysis on RE.

License GPL-3

Depends R (>= 3.6), SummarizedExperiment(>= 1.1.6), minfi (>= 1.22.0)

Imports readr, rtracklayer, graphics, stats, utils, methods, settings, BiocGenerics, S4Vectors, Biostrings, GenomicRanges, IRanges, Seqinfo, BiocParallel, doParallel, parallel, foreach, caret, kernlab, ranger, BSgenome, AnnotationHub, org.Hs.eg.db, impute, iterators

Suggests IlluminaHumanMethylation450kanno.ilmn12.hg19, IlluminaHumanMethylationEPICanno.ilm10b2.hg19, BSgenome.Hsapiens.UCSC.hg19, BSgenome.Hsapiens.UCSC.hg38, knitr, rmarkdown, minfiDataEPIC

Collate REMParcel-class.R REMProduct-class.R generics.R REMParcel-methods.R REMProduct-methods.R options.R utils.R REMPtools.R getGM12878.R grooMethy.R initREMP.R rempTemplate.R remp.R remprofile.R DemoData.R onAttach.R REMP-package.R

URL https://github.com/YinanZheng/REMP

BugReports https://github.com/YinanZheng/REMP/issues

Encoding UTF-8

LazyData true

biocViews DNAMethylation, Microarray, MethylationArray, Sequencing, GenomeWideAssociation, Epigenetics, Preprocessing, MultiChannel, TwoChannel, DifferentialMethylation, QualityControl, DataImport

RoxygenNote 7.1.2

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

git_branch RELEASE_3_23

git_last_commit 6b8fa77

git_last_commit_date 2026-04-28

Repository Bioconductor 3.23

Date/Publication 2026-11-11

Author Yinan Zheng [aut, cre],
Lei Liu [aut],
Wei Zhang [aut],
Warren Kibbe [aut],
Lifang Hou [aut, cph]

Maintainer Yinan Zheng <y-zheng@northwestern.edu>

Contents

REMP-package 2
Alu.hg19.demo 3
Alu.hg38.demo 4
fetchRefSeqGene 5
fetchRMSK 6
findRECpG 7
getBackend 9
getGM12878 9
GRannot 10
grooMethy 11
initREMP 13
remp 15
REMParcel-class 18
REMProduct-class 21
remprofile 24
rempTemplate 26
remp_options 28

Index **30**

`REMP-package` *Repetitive Element Methylation Prediction*

Description

Machine learning-based tools to predict DNA methylation of locus-specific repetitive elements (RE) by learning surrounding genetic and epigenetic information. These tools provide genomewide and single-base resolution of DNA methylation prediction on RE that are difficult to measure directly using array-based or sequencing-based platforms, which enables epigenome-wide association study (EWAS) and differentially methylated region (DMR) analysis on RE.

Overview - standard procedure

Step 1 Start out generating required dataset for prediction using `initREMP`. The datasets include RE information, RE-CpG (i.e. CpGs located in RE region) information, and gene annotation, which are maintained in a `REMParcel` object. It is recommended to save these generated data to the working directory so they can be used in the future.

Step 2 Clean Illumina methylation dataset using `grooMethy`. This function can help identify and fix abnormal values and automatically impute missing values, which are essential for downstream prediction.

Step 3 Run `remp` to predict genome-wide locus specific RE methylation.

Step 4 Use the built-in accessors and utilities in `REMProduct` object to get or refine the prediction results.

Author(s)

Yinan Zheng <y-zheng@northwestern.edu>, Lei Liu <lei.liu@northwestern.edu>, Wei Zhang <wei.zhang1@northwestern.edu>, Warren Kibbe <warren.kibbe@nih.gov>, Lifang Hou <l-hou@northwestern.edu>

Maintainer: Yinan Zheng <y-zheng@northwestern.edu>

References

Zheng Y, Joyce BT, Liu L, Zhang Z, Kibbe WA, Zhang W, Hou L. Prediction of genome-wide DNA methylation in repetitive elements. Nucleic Acids Res. 2017;45(15):8697-711. PubMed PMID: 28911103; PMCID: PMC5587781. http://dx.doi.org/10.1093/nar/gkx587.

`Alu.hg19.demo` *Subset of Alu genomic location dataset (hg19)*

Description

A `GRanges` dataset containing 500 Alu sequences that have CpGs profiled by both Illumina 450k and EPIC array. The variables are as follows:

Usage

```
Alu.hg19.demo
```

Format

A `GRanges` object.

Details

- seqnames: chromosome number
- ranges: hg19 genomic position
- strand: DNA strand
- swScore: Smith Waterman (SW) alignment score
- repName: Alu name
- repClass: Alu class
- repFamily: Alu family
- Index: internal index (meaningless for external use; not communicable between genome builds)

`Alu.hg19.demo` has the same format as the data object returned by `fetchRMSK`.

Value

A GRanges object with 500 ranges and 3 metadata columns.

Source

RepeatMasker database provided by package `AnnotationHub`

See Also

See `fetchRMSK` to obtain the complete Alu/L1 dataset.

`Alu.hg38.demo` *Subset of Alu genomic location dataset (hg38)*

Description

A `GRanges` dataset containing 500 Alu sequences that have CpGs profiled by both Illumina 450k and EPIC array. The variables are as follows:

Usage

```
Alu.hg38.demo
```

Format

A `GRanges` object.

Details

- seqnames: chromosome number
- ranges: hg38 genomic position
- strand: DNA strand
- swScore: Smith Waterman (SW) alignment score
- repName: Alu name
- repClass: Alu class
- repFamily: Alu family
- Index: internal index (meaningless for external use; not communicable between genome builds)

`Alu.hg38.demo` has the same format as the data object returned by `fetchRMSK`.

Value

A GRanges object with 500 ranges and 3 metadata columns.

Source

RepeatMasker database (hg38) is provided by UCSC database downloaded from http://hgdownload.cse.ucsc.edu/goldenpath.

See Also

See `fetchRMSK` to obtain the complete Alu/L1 dataset.

`fetchRefSeqGene` *Get RefSeq gene database*

Description

`fetchRefSeqGene` is used to obtain refSeq gene database provided by AnnotationHub (hg19) or UCSC web database (hg19/hg38).

Usage

```
fetchRefSeqGene(
  annotation.source = c("AH", "UCSC"),
  genome = c("hg19", "hg38"),
  mainOnly = FALSE,
  verbose = FALSE
)
```

Arguments

annotation.source	Character parameter. Specify the source of annotation databases, including the RefSeq Gene annotation database and RepeatMasker annotation database. If "AH", the database will be obtained from the AnnotationHub package. If "UCSC", the database will be downloaded from the UCSC website http://hgdownload.cse.ucsc.edu/goldenp The corresponding build ("hg19" or "hg38") will be specified in the parameter genome.
genome	Character parameter. Specify the build of human genome. Can be either "hg19" or "hg38". Note that if annotation.source == "AH", only hg19 database is available.
mainOnly	Logical parameter. See details.
verbose	Logical parameter. Should the function be verbose?

Details

When mainOnly = FALSE, only the transcript location information will be returned, Otherwise, a GRangesList object containing gene regions information will be added. Gene regions include: 2000 base pair upstream of the transcript start site ($tss)), 5'UTR ($fiveUTR)), coding sequence ($cds)), exon ($exon)), and 3'UTR ($threeUTR)). The index column is an internal index that is used to facilitate data referral, which is meaningless for external use.

Value

A single GRanges (for main refgene data) object or a list incorporating both GRanges object (for main refgene data) and GRangesList object (for gene regions data).

Examples

```
if (!exists("refgene.hg19"))
refgene.hg19 <- fetchRefSeqGene(annotation.source = "AH",
                                genome = "hg19",
                                verbose = TRUE)
refgene.hg19
```

---

fetchRMSK *Get RE database from RepeatMasker*

Description

fetchRMSK is used to obtain specified RE database from RepeatMasker Database provided by AnnotationHub.

Usage

```
fetchRMSK(
  REtype = c("Alu", "L1", "ERV"),
  annotation.source = c("AH", "UCSC"),
  genome = c("hg19", "hg38"),
  verbose = FALSE
)
```

Arguments

REtype	Type of RE. Currently "Alu", "L1", and "ERV" are supported.
annotation.source	Character parameter. Specify the source of annotation databases, including the RefSeq Gene annotation database and RepeatMasker annotation database. If "AH", the database will be obtained from the AnnotationHub package. If "UCSC", the database will be downloaded from the UCSC website http://hgdownload.cse.ucsc.edu/goldenp The corresponding build ("hg19" or "hg38") will be specified in the parameter genome.
genome	Character parameter. Specify the build of human genome. Can be either "hg19" or "hg38".
verbose	Logical parameter. Should the function be verbose?

Value

A GRanges object containing RE database. 'repName' column indicates the RE name; 'swScore' column indicates the SW score; 'Index' is an internal index for RE to facilitate data referral, which is meaningless for external use.

Examples

```
L1 <- fetchRMSK(REtype = "L1",
                annotation.source = "AH",
                genome = "hg19",
                verbose = TRUE)
L1
```

---

findRECpG *Find RE-CpG genomic location given RE ranges information*

Description

findRECpG is used to obtain RE-CpG genomic location data.

Usage

```
findRECpG(
  RE,
  REtype = c("Alu", "L1", "ERV"),
  genome = c("hg19", "hg38"),
  be = NULL,
  verbose = FALSE
)
```

Arguments

RE	A GRanges object of RE genomic location database. This can be obtained by fetchRMSK.
REtype	Type of RE. Currently "Alu", "L1", and "ERV" are supported.
genome	Character parameter. Specify the build of human genome. Can be either "hg19" or "hg38". User should make sure the genome build of RE is consistent with this parameter.
be	A BiocParallel object containing back-end information that is ready for parallel computing. This can be obtained by getBackend. If not specified, non-parallel mode is used.
verbose	logical parameter. Should the function be verbose?

Details

CpG site is defined as 5'-C-p-G-3'. It is reasonable to assume that the methylation status across all CpG/CpG dyads are concordant. Maintenance methyltransferase exhibits a preference for hemimethylated CpG/CpG dyads (methylated on one strand only). As a result, methyaltion status of CpG sites in both forward and reverse strands are usually consistent. Therefore, to accommodate the cytosine loci in both strands, the returned genomic ranges cover the 'CG' sequence with width of 2. The 'strand' information indicates the strand of the RE. Locating CpG sites in RE sequences can be computation intensive. It is recommanded to get more than one work in the backend for a faster running speed.

Value

A GRanges object containing identified RE-CpG genomic location data.

Examples

```
data(Alu.hg19.demo)
RE.CpG <- findRECpG(RE = Alu.hg19.demo,
                    REtype = "Alu",
                    genome = "hg19",
                    verbose = TRUE)
RE.CpG
```

getBackend *Get BiocParallel back-end*

Description

getBackend is used to obtain BiocParallel Back-end to allow parallel computing.

Usage

```
getBackend(ncore, BPPARAM = NULL, verbose = FALSE)
```

Arguments

ncore	Number of cores used for parallel computing. By default max number of cores available in the machine will be utilized. If ncore = 1, no parallel computing is allowed.
BPPARAM	An optional BiocParallelParam instance determining the parallel back-end to be used during evaluation. If not specified, default back-end in the machine will be used.
verbose	Logical parameter. Should the function be verbose?

Value

A BiocParallel object that can be used for parallel computing.

Examples

```
# Non-parallel mode
be <- getBackend(ncore = 1, verbose = TRUE)
be

# parallel mode (2 workers)
be <- getBackend(ncore = 2, verbose = TRUE)
be
```

getGM12878 *Get methylation data of HapMap LCL sample GM12878 profiled by Illumina 450k array or EPIC array*

Description

getGM12878 is used to obtain public available methylation profiling data of HapMap LCL sample GM12878.

Usage

```
getGM12878(arrayType = c("450k", "EPIC"), mapGenome = FALSE)
```

Arguments

arrayType	Illumina methylation array type. Currently "450k" and "EPIC" are supported. Default = "450k".
mapGenome	Logical parameter. If TRUE, function will return a GenomicRatioSet object instead of a link{RatioSet} object.

Details

Illumina 450k data were sourced and curated from ENCODE http://hgdownload.cse.ucsc.edu/goldenPath/hg19/encodeDCC/wgEncodeHaibMethyl450/wgEncodeHaibMethyl450Gm12878SitesRep1.bed.gz. Illumina EPIC data were obtained from data package minfiDataEPIC.

Value

A RatioSet or GenomicRatioSet containing beta value and M value of the methylation data.

Examples

```
## Not run:
# Get GM12878 methylation data (450k array)
if (!exists("GM12878_450k")) GM12878_450k <- getGM12878("450k")
GM12878_450k

## End(Not run)

# Get GM12878 methylation data (EPIC array)
if (!exists("GM12878_EPIC")) GM12878_EPIC <- getGM12878("EPIC")
GM12878_EPIC
```

GRannot *Annotate genomic ranges data with gene region information.*

Description

GRannot is used to annotate a GRanges dataset with gene region information using refseq gene database

Usage

```
GRannot(object.GR, refgene, symbol = FALSE, verbose = FALSE)
```

Arguments

object.GR	An GRanges object of a genomic location database.
refgene	A complete refGene annotation database returned by fetchRefSeqGene (with parameter mainOnly = FALSE).
symbol	Logical parameter. Should the annotation return gene symbol?
verbose	Logical parameter. Should the function be verbose?

Details

The annotated gene region information includes: protein coding gene (InNM), noncoding RNA gene (InNR), 2000 base pair upstream of the transcript start site (InTSS), 5'UTR (In5UTR), coding sequence (InCDS), exon (InExon), and 3'UTR (In3UTR). The intergenic and intron regions can then be represented by the combination of these region data. The number shown in these columns represent the row number or 'index' column in the main refgene database obtained by fetchRefSeqGene.

Value

A GRanges or a GRangesList object containing refSeq Gene database.

Examples

```
data(Alu.hg19.demo)
if (!exists("refgene.hg19"))
  refgene.hg19 <- fetchRefSeqGene(annotation.source = "AH",
                                  genome = "hg19",
                                  verbose = TRUE)
Alu.hg19.demo.refGene <- GRannot(Alu.hg19.demo, refgene.hg19, verbose = TRUE)
Alu.hg19.demo.refGene
```

---

grooMethy *Groom methylation data to fix potential data issues*

Description

grooMethy is used to automatically detect and fix data issues including zero beta value, missing value, and infinite value.

Usage

```
grooMethy(
  methyDat,
  Seq.GR = NULL,
  impute = TRUE,
  imputebyrow = TRUE,
  mapGenome = FALSE,
  verbose = FALSE
)
```

Arguments

methyDat	A RatioSet, GenomicRatioSet, DataFrame, data.table, data.frame, or matrix of Illumina BeadChip methylation data (450k or EPIC array) or Illumina methylation percentage estimates by sequencing. If the data are prepared as a data.frame or alike format, for Illumina array data, please make sure there is a column or row names are available to indicate the Illumina probe names (i.e. cg00000029); for sequencing methylation data, please provide the corresponding CpG location information in Seq.GR.
Seq.GR	A GRanges object containing genomic locations of the CpGs profiled by sequencing platforms. This parameter should not be NULL if the input methylation data methyDat are obtained by sequencing platform. The order of Seq.GR should match the order of methyDat. Note that the genomic location can be in either hg19 or hg38 build. See details.
impute	If TRUE, K-Nearest Neighbouring imputation will be applied to fill the missing values. Default = TRUE. See Details.
imputebyrow	If TRUE, missing values will be imputed using similar values in row (i.e., across samples); if FALSE, missing values will be imputed using similar values in column (i.e., across CpGs). Default is TRUE.
mapGenome	Logical parameter. If TRUE, function will return a GenomicRatioSet object instead of a RatioSet. This function is not applicable for sequencing data.
verbose	Logical parameter. Should the function be verbose?

Details

For methylation data in beta value, if zero/one value exists, the logit transformation from beta to M value will produce infinite value. Therefore, zero/one beta value will be replaced with the smallest non-zero beta/largest non-one beta value found in the dataset. grooMethy can also handle missing value (i.e. NA or NaN) using KNN imputation (see impute.knn). The infinite value will be also treated as missing value for imputation. If the original dataset is in beta value, grooMethy will first transform it to M value before imputation is carried out. If the imputed value is out of the original range (which is possible when imputebyrow = FALSE), mean value will be used instead. Warning: imputed values for multimodal distributed CpGs (across samples) may not be correct. Please check package ENmix to identify the CpGs with multimodal distribution. Please note that grooMethy is also embedded in remp so the user can run remp directly without explicitly running grooMethy. For sequencing methylation data, please specify the genomic location of CpGs in a GenomicRanges object and specify it in Seq.GR. For an example of Seq.GR, Please run minfi::getLocations(IlluminaHumanMethylation450kanno.ilmn12.hg19) (the row names of the CpGs in Seq.GR can be NULL). The user should make sure the genome build of Seq.GR match the build specified in genome parameter of function initREMP and remprofile (default is "hg19").

Value

A RatioSet or GenomicRatioSet containing beta value and M value of the methylation data.

Examples

```
# Get GM12878 methylation data (450k array)
```

```
if (!exists("GM12878_450k")) GM12878_450k <- getGM12878("450k")
GM12878_450k <- grooMethy(GM12878_450k, verbose = TRUE)

# Also works if data input is a matrix
grooMethy(minfi::getBeta(GM12878_450k), verbose = TRUE)
```

---

initREMP *RE Annotation Database Initialization*

Description

initREMP is used to initialize annotation database for RE methylation prediction. Three RE types in human, Alu element (Alu), LINE-1 (L1), and endogenous retrovirus (ERV) are available.

Usage

```
initREMP(
  arrayType = c("450k", "EPIC", "Sequencing"),
  REtype = c("Alu", "L1", "ERV"),
  annotation.source = c("AH", "UCSC"),
  genome = c("hg19", "hg38"),
  RE = NULL,
  Seq.GR = NULL,
  ncore = NULL,
  BPPARAM = NULL,
  export = FALSE,
  work.dir = tempdir(),
  verbose = FALSE
)
```

Arguments

arrayType	Illumina methylation array type. Currently "450k", "EPIC", and "Sequencing" are supported. Default = "450k".
REtype	Type of RE. Currently "Alu", "L1", and "ERV" are supported.
annotation.source	Character parameter. Specify the source of annotation databases, including the RefSeq Gene annotation database and RepeatMasker annotation database. If "AH", the database will be obtained from the AnnotationHub package. If "UCSC", the database will be downloaded from the UCSC website http://hgdownload.cse.ucsc.edu/goldenp The corresponding build ("hg19" or "hg38") can be specified in the parameter genome.
genome	Character parameter. Specify the build of human genome. Can be either "hg19" or "hg38". Note that if annotation.source == "AH", only hg19 database is available.

RE	A GRanges object containing user-specified RE genomic location information. If NULL, the function will retrive RepeatMasker RE database from AnnotationHub (build hg19) or download the database from UCSC website (build hg19/hg38).
Seq.GR	A GRanges object containing genomic locations of the CpGs profiled by sequencing platforms. This parameter should not be NULL if arrayType == 'Sequencing'. Note that the genomic location can be in either hg19 or hg38 build. See details.
ncore	Number of cores used for parallel computing. By default max number of cores available in the machine will be utilized. If ncore = 1, no parallel computing is allowed.
BPPARAM	An optional BiocParallelParam instance determining the parallel back-end to be used during evaluation. If not specified, default back-end in the machine will be used.
export	Logical. Should the returned REMParcel object be saved to local machine? See Details.
work.dir	Path to the directory where the generated data will be saved. Valid when export = TRUE. If not specified and export = TRUE, temporary directory tempdir() will be used.
verbose	Logical parameter. Should the function be verbose?

Details

Currently, we support two major types of RE in the human genome, Alu and L1. The main purpose of initREMP is to generate and annotate CpG/RE data using the refSeq Gene (hg19) annotation database (provided by AnnotationHub). These annotation data are crucial to RE methylation prediction in remp. Once generated, the data can be reused in the future (data can be very large). Therefore, we recommend the user to save the output from initREMP to the local machine, so that user only need to run this function once as long as there is no change to the RE database. To minimize the size of the resulting data file, the generated annotation data are only for REs that contain RE-CpGs with neighboring profiled CpGs. By default, the neighboring CpGs are confined within 1200 bp flanking window. This window size can be modified using remp_options. Note that the refSeq Gene database from UCSC is dynamic (updated periodically) and reflecting the latest knowledge of gene, whereas the database from AnnotationHub is static and classic. Using different sources will have a slight impact on the prediction results of RE methylation and gene annotation of final results. For sequencing methylation data, please specify the genomic location of CpGs in a GenomicRanges object and specify it in Seq.GR. For an example of Seq.GR, Please run minfi::getLocations(IlluminaHumanMethylation450kanno.ilmn12.hg19) (the row names of the CpGs in Seq.GR can be NULL). The user should make sure the genome build of Seq.GR match the build specified in genome parameter (default is "hg19").

Value

An REMParcel object containing data needed for RE methylation prediction.

See Also

See remp for RE methylation prediction.

Examples

```
if (!exists("remparcel")) {
  data(Alu.hg19.demo)
  remparcel <- initREMP(arrayType = "450k",
                        REtype = "Alu",
                        annotation.source = "AH",
                        genome = "hg19",
                        RE = Alu.hg19.demo,
                        ncore = 1,
                        verbose = TRUE)
}
```

`remp` *Repetitive element methylation prediction*

Description

`remp` is used to predict genomewide methylation levels of locus-specific repetitive elements (RE). Two major RE types in human, Alu element (Alu) and LINE-1 (L1) are available.

Usage

```
remp(
  methyDat = NULL,
  REtype = c("Alu", "L1", "ERV"),
  Seq.GR = NULL,
  parcel = NULL,
  work.dir = tempdir(),
  win = 1000,
  method = c("rf", "xgbTree", "svmLinear", "svmRadial", "naive"),
  autoTune = TRUE,
  param = NULL,
  seed = NULL,
  ncore = NULL,
  BPPARAM = NULL,
  verbose = FALSE
)
```

Arguments

`methyDat`	A `RatioSet`, `GenomicRatioSet`, `DataFrame`, `data.table`, `data.frame`, or `matrix` of Illumina BeadChip methylation data (450k or EPIC array) or Illumina methylation percentage estimates by sequencing. See Details. Alternatively, user can also specify a pre-built data template (see `rempTemplate`). `remp` to carry out the prediction. See `rempTemplate`. With template specified, `methyDat`, `REtype`, `parcel`, and `work.dir` can be skipped.

REtype	Type of RE. Currently "Alu", "L1", and "ERV" are supported. If NULL, the type of RE will be extracted from parcel.
Seq.GR	A GRanges object containing genomic locations of the CpGs profiled by sequencing platforms. This parameter should not be NULL if the input methylation data methyDat are obtained by sequencing. Note that the genomic location can be in either hg19 or hg38 build. See details in initREMP.
parcel	An REMParcel object containing necessary data to carry out the prediction. If NULL, REtype must specify a type of RE so that the function can search the .rds data file in work.dir exported by initREMP (with export = TRUE) or saveParcel.
work.dir	Path to the directory where the annotation data generated by initREMP are saved. Valid when the argument parcel is missing. If not specified, temporary directory tempdir() will be used. If specified, the directory path has to be the same as the one specified in initREMP or in saveParcel.
win	An integer specifying window size to confine the upstream and downstream flanking region centered on the predicted CpG in RE for prediction. Default = 1000. See Details.
method	Name of model/approach for prediction. Currently "rf" (Random Forest), "xgbTree" (Extreme Gradient Boosting), "svmLinear" (SVM with linear kernel), "svmRadial" (SVM with radial kernel), and "naive" (carrying over methylation values of the closest CpG site) are available. Default = "rf" (Random Forest). See Details.
autoTune	Logical parameter. If TRUE, a 3-time repeated 5-fold cross validation will be performed to determine the best model parameter. If FALSE, the param option (see below) must be specified. Default = TRUE. Auto-tune will be disabled using Random Forest. See Details.
param	A list specifying fixed model tuning parameter(s) (not applicable for Random Forest, see Details). For Extreme Gradient Boosting, param list must contain '$nrounds', '$max_depth', '$eta', '$gamma', '$colsample_bytree', '$min_child_weight', and '$subsample'. See xgbTree in package caret. For SVM, param list must contain '$C' (cost) for linear kernel or '$sigma' and '$C' for radial basis function kernel. This parameter is valid only when autoTune = FALSE.
seed	Random seed for Random Forest model for reproducible prediction results. Default is NULL, which generates a seed.
ncore	Number of cores used for parallel computing. By default, max number of cores available in the machine will be utilized. If ncore = 1, no parallel computing is allowed.
BPPARAM	An optional BiocParallelParam instance determining the parallel back-end to be used during evaluation. If not specified, default back-end in the machine will be used.
verbose	Logical parameter. Should the function be verbose?

Details

Before running remp, user should make sure the methylation data have gone through proper quality control, background correction, and normalization procedures. Both beta value and M value are

allowed. Rows represents probes and columns represents samples. For array data, please make sure to have row names that specify the Illumina probe ID (i.e. cg00000029). For sequencing data, please provide the genomic location of CpGs in a GRanges obejct and specify it using Seq.GR parameter. win = 1000 is based on previous findings showing that neighboring CpGs are more likely to be co-modified within 1000 bp. User can specify narrower window size for slight improvement of prediction accuracy at the cost of less predicted RE. Window size greater than 1000 is not recommended as the machine learning models would not be able to learn much userful information for prediction but introduce noise. Random Forest model (method = "rf") is recommented as it offers more accurate prediction and it also enables prediction reliability functionality. Prediction reliability is estimated by conditional standard deviation using Quantile Regression Forest. Please note that if parallel computing is allowed, parallel Random Forest (powered by package ranger) will be used automatically. The performance of Random Forest model is often relatively insensitive to the choice of mtry. Therefore, auto-tune will be turned off using Random Forest and mtry will be set to one third of the total number of predictors. For SVM, if autoTune = TRUE, preset tuning parameter search grid can be access and modified using remp_options.

Value

A REMProduct object containing predicted RE methylation results.

See Also

See initREMP to prepare necessary annotation database before running remp.

Examples

```
# Obtain example Illumina example data (450k)
if (!exists("GM12878_450k"))
  GM12878_450k <- getGM12878("450k")

# Make sure you have run 'initREMP' first. See ?initREMP.
if (!exists("remparcel")) {
  data(Alu.hg19.demo)
  remparcel <- initREMP(arrayType = "450k",
                        REtype = "Alu",
                        annotation.source = "AH",
                        genome = "hg19",
                        RE = Alu.hg19.demo,
                        ncore = 1,
                        verbose = TRUE)
}

# With data template pre-built. See ?rempTemplate.
if (!exists("template"))
  template <- rempTemplate(GM12878_450k,
                           parcel = remparcel,
                           win = 1000,
                           verbose = TRUE)

# Run remp with pre-built template:
remp.res <- remp(template, ncore = 1)
```

```
# Or run remp without pre-built template (identical results):
## Not run:
  remp.res <- remp(GM12878_450k,
                   REtype = "Alu",
                   parcel = remparcel,
                   ncore = 1,
                   verbose = TRUE)

## End(Not run)

remp.res
details(remp.res)
rempB(remp.res) # Methylation data (beta value)

# Extract CpG location information.
# This accessor is inherit from class 'RangedSummarizedExperiment')
rowRanges(remp.res)

# RE annotation information
rempAnnot(remp.res)

# Add gene annotation
remp.res <- decodeAnnot(remp.res, type = "symbol")
rempAnnot(remp.res)

# (Recommended) Trim off less reliable prediction
remp.res <- rempTrim(remp.res)

# Obtain RE-level methylation (aggregate by mean)
remp.res <- rempAggregate(remp.res)
rempB(remp.res) # Methylation data (beta value)

# Extract RE location information
rowRanges(remp.res)

# Density plot across predicted RE
remplot(remp.res)
```

`REMParcel-class` *REMParcel instances*

Description

`REMParcel` is a container class to organize required datasets for RE methylation prediction generated from `initREMP` and used in `remp`.

Usage

```
REMParcel(
  REtype = "Unknown",
  genome = "Unknown",
  platform = "Unknown",
  RefGene = GRanges(),
  RE = GRanges(),
  RECpG = GRanges(),
  ILMN = GRanges()
)

getParcelInfo(object)

getRefGene(object)

getRE(object)

getRECpG(object)

getILMN(object, ...)

saveParcel(object, ...)

## S4 method for signature 'REMParcel'
saveParcel(object, work.dir = tempdir(), verbose = FALSE, ...)

## S4 method for signature 'REMParcel'
getParcelInfo(object)

## S4 method for signature 'REMParcel'
getRefGene(object)

## S4 method for signature 'REMParcel'
getRE(object)

## S4 method for signature 'REMParcel'
getRECpG(object)

## S4 method for signature 'REMParcel'
getILMN(object, REonly = FALSE)
```

Arguments

REtype	Type of RE ("Alu", "L1", or "ERV").
genome	Specify the build of human genome. Can be either "hg19" or "hg38".
platform	Illumina methylation profiling platform ("450k" or "EPIC").
RefGene	refSeq gene annotation data, which can be obtained by fetchRefSeqGene.

RE	Annotated RE genomic range data, which can be obtained by fetchRMSK and annotated by GRannot.
RECpG	Genomic range data of annotated CpG site identified in RE DNA sequence, which can be obtained by findRECpG and annotated by GRannot.
ILMN	Illumina CpG probe genomic range data.
object	A REMParcel object.
...	For saveParcel: other parameters to be passed to the saveRDS method. See saveRDS.
work.dir	For saveParcel: path to the directory where the generated data will be saved. If not specified, temporary directory tempdir() will be used.
verbose	For saveParcel: logical parameter. Should the function be verbose?
REonly	For getILMN: see Accessors.

Value

An object of class REMParcel for the constructor.

Accessors

getParcelInfo(object) Return data type, RE type, and flanking window size information of the parcel.

getRefGene(object) Return RefSeq gene annotation data.

getRE(object) Return RE genomic location data for prediction (annotated by refSeq gene database).

getRECpG(object) Return RE-CpG genomic location data for prediction.

getILMN(object, REonly = FALSE) Return Illumina CpG probe genomic location data for prediction (annotated by refSeq gene database). If REonly = TRUE, only probes within RE region are returned.

Utilities

saveParcel(object, work.dir = tempdir(), verbose = FALSE, ...) Save the object to local machine.

Examples

```
showClass("REMParcel")
```

REMProduct-class *REMProduct instances*

Description

Class REMProduct is to maintain RE methylation prediction results. REMProduct inherits Bioconductor's RangedSummarizedExperiment class.

Usage

```
REMProduct(
  REtype = "Unknown",
  genome = "Unknown",
  platform = "Unknown",
  win = "Unknown",
  predictModel = "Unknown",
  QCModel = "Unknown",
  rempM = NULL,
  rempB = NULL,
  rempQC = NULL,
  cpgRanges = GRanges(),
  sampleInfo = DataFrame(),
  REannotation = GRanges(),
  RECpG = GRanges(),
  regionCode = DataFrame(),
  refGene = GRanges(),
  varImp = DataFrame(),
  REStats = DataFrame(),
  GeneStats = DataFrame(),
  Seed = NULL
)

rempM(object)

rempB(object)

rempQC(object)

rempAnnot(object)

rempImp(object)

rempStats(object)

remplot(object, ...)

details(object)
```

```
decodeAnnot(object, ...)

rempTrim(object, ...)

rempAggregate(object, ...)

rempCombine(object1, object2)

## S4 method for signature 'REMProduct'
rempM(object)

## S4 method for signature 'REMProduct'
rempB(object)

## S4 method for signature 'REMProduct'
rempQC(object)

## S4 method for signature 'REMProduct'
rempImp(object)

## S4 method for signature 'REMProduct'
rempAnnot(object)

## S4 method for signature 'REMProduct'
rempStats(object)

## S4 method for signature 'REMProduct'
remplot(object, type = c("individual", "overall"), ...)

## S4 method for signature 'REMProduct'
details(object)

## S4 method for signature 'REMProduct'
decodeAnnot(object, type = c("symbol", "entrez"), ncore = 1, BPPARAM = NULL)

## S4 method for signature 'REMProduct'
rempTrim(object, threshold = 1.7, missingRate = 0.2)

## S4 method for signature 'REMProduct'
rempAggregate(object, NCpG = 2, ncore = 1, BPPARAM = NULL)

## S4 method for signature 'REMProduct,REMProduct'
rempCombine(object1, object2)
```

Arguments

REtype	Type of RE ("Alu", "L1", or "ERV").

genome	Specify the build of human genome. Can be either "hg19" or "hg38".
platform	Illumina methylation profiling platform ("450k" or "EPIC").
win	Flanking window size of the predicting RE-CpG.
predictModel	Name of the model used for prediction.
QCModel	Name of the model used for prediction quality evaluation.
rempM	Predicted methylation level in M value.
rempB	Predicted methylation level in beta value (optional).
rempQC	Prediction quality scores, which is available only when Random Forest model is used in remp.
cpgRanges	Genomic ranges of the predicting RE-CpG.
sampleInfo	Sample information.
REannotation	Annotation data for the predicting RE.
RECpG	Annotation data for the RE-CpG profiled by Illumina platform.
regionCode	Internal index code defined in refGene for gene region indicators.
refGene	refSeq gene annotation data, which can be obtained by fetchRefSeqGene.
varImp	Importance of the predictors.
REStats	RE coverage statistics, which is internally generated in remp.
GeneStats	Gene coverage statistics, which is internally generated in remp.
Seed	Random seed for Random Forest model for reproducible prediction results.
object	A REMProduct object.
...	For plot: graphical parameters to be passed to the plot method.
object1	A REMProduct object.
object2	A REMProduct object.
type	For plot and decodeAnnot: see Utilities.
ncore	For decodeAnnot and rempAggregate: number of cores used for parallel computing. By default no parallel computing is allowed (ncore = 1).
BPPARAM	For decodeAnnot and rempAggregate: an optional BiocParallelParam instance determining the parallel back-end to be used during evaluation. If not specified, default back-end in the machine will be used.
threshold	For rempTrim: see Utilities.
missingRate	For rempTrim: see Utilities.
NCpG	For rempAggregate: see Utilities.

Value

An object of class REMProduct for the constructor.

Accessors

`rempM(object)` Return M value of the prediction.

`rempB(object)` Return beta value of the prediction.

`rempQC(object)` Return prediction quality scores.

`rempImp(object)` Return relative importance of predictors.

`rempStats(object)` Return RE and gene coverage statistics.

`rempAnnot(object)` Return annotation data for the predicted RE.

Utilities

`remplot(object, type = c("individual", "overall"), ...)` Make a density plot of predicted methylation (beta values) in the `REMProduct` object. If `type = "individual"`, density curves will be plotted for each of the samples; If `type = "overall"`, one density curve of the mean methylation level across the samples will be plotted. Default `type = "individual"`.

`details(object)` Display detailed descriptive statistics of the predicion results.

`decodeAnnot(object, type = c("symbol", "entrez")), ncore = NULL, BPPARAM = NULL` Decode the RE annotation data by Gene Symbol (when `type = "Symbol"`) or Entrez Gene (when `type = "Entrez"`).Default `type = "Symbol"`. Annotation data are provided by `org.Hs.eg.db`.

`rempTrim(object, threshold = 1.7, missingRate = 0.2)` Any predicted CpG values with quality score < threshold (default = 1.7, specified by `threshold = 1.7`) will be replaced with NA. CpGs contain more than missingRate * 100 rate across samples will be discarded. Relavant summary statistics will be re-evaluated.

`rempAggregate(object, NCpG = 2, ncore = NULL, BPPARAM = NULL)` Aggregate the predicted RE-CpG methylation by RE using mean. To ensure the reliability of the aggregation, by default only RE with at least 2 predicted CpG sites (specified by `NCpG = 2`) will be aggregated.

`rempCombine(object1, object2)` Combine two `REMProduct` objects by column.

Examples

```
showClass("REMProduct")
```

---

remprofile *Extract DNA methylation data profiled in RE*

Description

`remprofile` is used to extract profiled methylation of CpG sites in RE.

Usage

```
remprofile(
  methyDat,
  REtype = c("Alu", "L1", "ERV"),
  annotation.source = c("AH", "UCSC"),
  genome = c("hg19", "hg38"),
  Seq.GR = NULL,
  RE = NULL,
  impute = FALSE,
  imputebyrow = TRUE,
  verbose = FALSE
)
```

Arguments

methyDat	A RatioSet, GenomicRatioSet, DataFrame, data.table, data.frame, or matrix of Illumina BeadChip methylation data (450k or EPIC array) or Illumina methylation percentage estimates by sequencing.
REtype	Type of RE. Currently "Alu", "L1", and "ERV" are supported.
annotation.source	Character parameter. Specify the source of annotation databases, including the RefSeq Gene annotation database and RepeatMasker annotation database. If "AH", the database will be obtained from the AnnotationHub package. If "UCSC", the database will be downloaded from the UCSC website http://hgdownload.cse.ucsc.edu/goldenp The corresponding build ("hg19" or "hg38") will be specified in the parameter genome.
genome	Character parameter. Specify the build of human genome. Can be either "hg19" or "hg38". For 450k/EPIC array, "hg19" is used more often while specifying "hg38" will lift over the Illumina CpG probe location to build "hg38". For sequencing data, please make sure the specified genome build is consistent with the actual genome build of Seq.GR.
Seq.GR	A GRanges object containing genomic locations of the CpGs profiled by sequencing platforms. This parameter should not be NULL if the input methylation data methyDat are obtained by sequencing. Note that the genomic location can be in either hg19 or hg38 build. The user should make sure the parameter genome is correctly specified.
RE	A GRanges object containing user-specified RE genomic location information. If NULL, the function will retrive RepeatMasker RE database from AnnotationHub (build hg19) or download the database from UCSC website (build hg19/hg38).
impute	Parameter used by grooMethy. If TRUE, K-Nearest Neighbouring imputation will be applied to fill the missing values. Default = FALSE.
imputebyrow	Parameter used by grooMethy. If TRUE, missing values will be imputed using similar values in row (i.e., across samples); if FALSE, missing values will be imputed using similar values in column (i.e., across CpGs). Default is TRUE.
verbose	Logical parameter. Should the function be verbose?

Value

A REMProduct object containing profiled RE methylation results.

Examples

```
data(Alu.hg19.demo)
if (!exists("GM12878_450k")) GM12878_450k <- getGM12878("450k")
remprofile.res <- remprofile(GM12878_450k,
                             REtype = "Alu",
                             annotation.source = "AH",
                             genome = "hg19",
                             RE = Alu.hg19.demo,
                             verbose = TRUE)
details(remprofile.res)
rempB(remprofile.res) # Methylation data (beta value)

remprofile.res <- rempAggregate(remprofile.res)
details(remprofile.res)
rempB(remprofile.res) # Methylation data (beta value)
```

---

rempTemplate *Prepare data template for REMP*

Description

rempTemplate is used to build a set of data templates for prediction. The data templates include RE-CpGs and their methylation data for model training, neighboring CpGs of RE-CpGs and their methylation data for model prediction, and other necessary information about the prediction. This function is useful when one needs to experiment different tunning parameters so that these pre-built data templates can be re-used and substaintially improve efficiency.

Usage

```
rempTemplate(
  methyDat = NULL,
  Seq.GR = NULL,
  parcel = NULL,
  win = 1000,
  verbose = FALSE
)
```

Arguments

methyDat	A RatioSet, GenomicRatioSet, DataFrame, data.table, data.frame, or matrix of Illumina BeadChip methylation data (450k or EPIC array) or Illumina methylation percentage estimates by sequencing.

Seq.GR	A GRanges object containing genomic locations of the CpGs profiled by sequencing platforms. This parameter should not be NULL if the input methylation data methyDat are obtained by sequencing. Note that the genomic location can be in either hg19 or hg38 build, but must be consistent with the build as parcel. See details in initREMP.
parcel	An REMParcel object containing necessary data to carry out the prediction. If NULL, the function will search the .rds data file in work.dir exported by initREMP (with export = TRUE) or saveParcel.
win	An integer specifying window size to confine the upstream and downstream flanking region centered on the predicted CpG in RE for prediction. Default = 1000.
verbose	Logical parameter. Should the function be verbose?

Value

A template object containing a GRanges object of neifhboring CpGs of RE-CpGs to be predicted ($NBCpG_GR) and their methylation dataset matrix ($NBCpG_methyDat); a GRanges object of RE-CpGs for model training ($RECpG_GR) and their methylation dataset matrix ($RECpG_methyDat); GRanges objects of RefSeq Gene database ($refgene) and RE ($RE) that are extracted from the parcel input; a string of RE type ($REtype) and a string of methylation platform ($arrayType). Note: the subset operator [] is supported.

Examples

```
if (!exists("GM12878_450k"))
  GM12878_450k <- getGM12878("450k")
if (!exists("remparcel")) {
  data(Alu.hg19.demo)
  remparcel <- initREMP(arrayType = "450k",
                        REtype = "Alu",
                        annotation.source = "AH",
                        genome = "hg19",
                        RE = Alu.hg19.demo,
                        ncore = 1,
                        verbose = TRUE)
}

template <- rempTemplate(GM12878_450k,
                         parcel = remparcel,
                         win = 1000,
                         verbose = TRUE)
template

## To make a subset
template[1]
```

`remp_options` *Set or get options for REMP package*

Description

Tools to manage global setting options for REMP package.

Usage

```
remp_options(...)

remp_reset()
```

Arguments

`...`	Option names to retrieve option values or `[key]=[value]` pairs to set options.

Value

`NULL`

Supported options

The following options are supported

`.default.AluFamily.grep` Regular expression for 'grep' to extract Alu family to be included in the prediction.

`.default.L1Family.grep` Regular expression for 'grep' to extract L1 family to be included in the prediction.

`.default.ERVFamily.grep` Regular expression for 'grep' to extract ERV family to be included in the prediction.

`.default.chr` List of human chromosome.

`.default.GM12878.450k.URL` URL to download GM12878 450k methylation profiling data.

`.default.RMSK.hg19.URL` URL to download RepeatMasker database in hg19 genome.

`.default.RMSK.hg38.URL` URL to download RepeatMasker database in hg38 genome.

`.default.refGene.hg19.URL` URL to download refSeq gene database in hg19 genome.

`.default.refGene.hg38.URL` URL to download refSeq gene database in hg38 genome.

`.default.AH.repeatmasker.hg19` `AnnotationHub` data ID linked to RepeatMasker annotation database (Mar 2020, build hg19).

`.default.AH.repeatmasker.hg38` `AnnotationHub` data ID linked to RepeatMasker annotation database (Sep 2021, build hg38).

`.default.AH.refgene.hg19` `AnnotationHub` data ID linked to refSeq gene database (build hg19)

`.default.AH.hg38ToHg19.over.chain` `AnnotationHub` hg38 to hg19 liftover chain data ID.

`.default.AH.hg19ToHg38.over.chain` `AnnotationHub` hg19 to hg38 liftover chain data ID.

`.default.TSS.upstream` Define the upstream range of transcription start site region.

`.default.TSS.downstream` Define the downstream range of transcription start site region.

`.default.max.flankWindow` Define the max size of the flanking window surrounding the predicted RE-CpG.

`.default.27k.total.probes` Total number of probes designed in Illumina 27k array.

`.default.450k.total.probes` Total number of probes designed in Illumina 450k array.

`.default.epic.total.probes` Total number of probes designed in Illumina EPIC array.

`.default.450k.annotation` A character string associated with the Illumina 450k array annotation dataset.

`.default.epic.annotation` A character string associated with the Illumina EPIC array annotation dataset.

`.default.genomicRegionColNames` Define the names of the genomic regions for prediction.

`.default.predictors` Define the names of predictors for RE methylation prediction.

`.default.svmLinear.tune` Define the default `C` (Cost) parameter for Support Vector Machine (SVM) using linear kernel.

`.default.svmRadial.tune` Define the default parameters (`C` and `sigma`) for SVM using Radial basis function kernel.

`.default.xgbTree.tune` Define the default parameters (`nrounds`, `eta`, `max_depth`, `gamma`, `colsample_bytree`, `min_child_weight`, and `subsample`) for Extreme Gradient Boosting.

Examples

```
# Display all default settings
remp_options()

# Display a specified setting
remp_options(".default.max.flankWindow")

# Change default maximum flanking window size to 2000
remp_options(.default.max.flankWindow = 2000)

# Reset all options
remp_reset()
```

Index

∗ **datasets**
  Alu.hg19.demo, 3
  Alu.hg38.demo, 4
∗ **package**
  REMP-package, 2

Alu.hg19.demo, 3
Alu.hg38.demo, 4
AnnotationHub, *14*, *25*

BiocParallel, *8, 9*
BiocParallelParam, *14*, *16*, *23*

DataFrame, *12*, *15*, *25, 26*
decodeAnnot *(*REMProduct-class*)*, 21
decodeAnnot,REMProduct-method
  *(*REMProduct-class*)*, 21
details *(*REMProduct-class*)*, 21
details,REMProduct-method
  *(*REMProduct-class*)*, 21

fetchRefSeqGene, 5, *11*, *19*, *23*
fetchRMSK, *4, 5*, 6, *8*, *20*
findRECpG, 7, *20*

GenomicRatioSet, *10*, *12*, *15*, *25, 26*
getBackend, *8*, 9
getGM12878, 9
getILMN *(*REMParcel-class*)*, 18
getILMN,REMParcel-method
  *(*REMParcel-class*)*, 18
getParcelInfo *(*REMParcel-class*)*, 18
getParcelInfo,REMParcel-method
  *(*REMParcel-class*)*, 18
getRE *(*REMParcel-class*)*, 18
getRE,REMParcel-method
  *(*REMParcel-class*)*, 18
getRECpG *(*REMParcel-class*)*, 18
getRECpG,REMParcel-method
  *(*REMParcel-class*)*, 18
getRefGene *(*REMParcel-class*)*, 18
getRefGene,REMParcel-method
  *(*REMParcel-class*)*, 18
GRanges, *3, 4*, *6–8*, *10–12*, *14*, *16, 17*, *25*, *27*
GRangesList, *6*, *11*
GRannot, 10, *20*
grooMethy, *3*, 11, *25*

impute.knn, *12*
initREMP, *3*, *12*, 13, *16–18*, *27*

org.Hs.eg.db, *24*

ranger, *17*
RatioSet, *10*, *12*, *15*, *25, 26*
REMP *(*REMP-package*)*, 2
remp, *3*, *12*, *14*, 15, *18*, *23*
REMP-package, 2
remp_options, *14*, *17*, 28
remp_reset *(*remp_options*)*, 28
rempAggregate *(*REMProduct-class*)*, 21
rempAggregate,REMProduct-method
  *(*REMProduct-class*)*, 21
rempAnnot *(*REMProduct-class*)*, 21
rempAnnot,REMProduct-method
  *(*REMProduct-class*)*, 21
REMParcel, *3*, *14*, *16*, *27*
REMParcel *(*REMParcel-class*)*, 18
REMParcel-class, 18
rempB *(*REMProduct-class*)*, 21
rempB,REMProduct-method
  *(*REMProduct-class*)*, 21
rempCombine *(*REMProduct-class*)*, 21
rempCombine,REMProduct,REMProduct-method
  *(*REMProduct-class*)*, 21
rempImp *(*REMProduct-class*)*, 21
rempImp,REMProduct-method
  *(*REMProduct-class*)*, 21
remplot *(*REMProduct-class*)*, 21
remplot,REMProduct-method
  *(*REMProduct-class*)*, 21

rempM *(*REMProduct-class*)*, 21
rempM,REMProduct-method
*(*REMProduct-class*)*, 21
rempQC *(*REMProduct-class*)*, 21
rempQC,REMProduct-method
*(*REMProduct-class*)*, 21
REMProduct, *3*, *17*, *26*
REMProduct *(*REMProduct-class*)*, 21
REMProduct-class, 21
remprofile, *12*, 24
rempStats *(*REMProduct-class*)*, 21
rempStats,REMProduct-method
*(*REMProduct-class*)*, 21
rempTemplate, *15*, 26
rempTrim *(*REMProduct-class*)*, 21
rempTrim,REMProduct-method
*(*REMProduct-class*)*, 21

saveParcel, *16*, *27*
saveParcel *(*REMParcel-class*)*, 18
saveParcel,REMParcel-method
*(*REMParcel-class*)*, 18
saveRDS, *20*