Package 'MethReg'

March 30, 2021

Type Package

Title Assessing the regulatory potential of DNA methylation regions or sites on gene transcription

Version 1.0.0

Description Epigenome-wide association studies (EWAS) detects a large number of DNA methylation differences, often hundreds of differentially methylated regions and thousands of CpGs, that are significantly associated with a disease, many are located in non-coding regions.
Therefore, there is a critical need to better understand the functional impact of these CpG methylations and to further prioritize the significant changes.
MethReg is an R package for integrative modeling of DNA methylation, target gene expression and transcription factor binding sites data, to systematically identify and rank functional CpG methylations.
MethReg evaluates, prioritizes and annotates CpG sites with high regulatory potential using matched methylation and gene expression data, along with external TF-target interaction databases based on manually curation, ChIP-seq experiments or gene regulatory network analysis.

License GPL-3

Encoding UTF-8

LazyData true

Imports dplyr, plyr, GenomicRanges, SummarizedExperiment, DelayedArray, ggplot2, ggpubr, tibble, tidyr, S4Vectors, sesameData, stringr, readr, methods, stats, Matrix, MASS, rlang, pscl, IRanges, sfsmisc, progress

Suggests rmarkdown, BiocStyle, testthat (>= 2.1.0), parallel, downloader, R.utils, doParallel, reshape2, JASPAR2020, TFBSTools, motifmatchr, matrixStats, biomaRt, dorothea, viper, stageR, BiocFileCache, png, htmltools, knitr, jpeg

VignetteBuilder knitr

BugReports https://github.com/TransBioInfoLab/MethReg/issues/

RoxygenNote 7.1.1

Depends R (>= 4.0)

biocViews MethylationArray, Regression, GeneExpression, Epigenetics, GeneTarget, Transcription

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

git_branch RELEASE_3_12

git_last_commit 2932aa8

git_last_commit_date 2020-10-27

Date/Publication 2021-03-29

Author Tiago Silva [aut, cre] (<https://orcid.org/0000-0003-1343-6850>), Lily Wang [aut]

Maintainer Tiago Silva <tiagochst@gmail.com>

R topics documented:

`clinical` *TCGA-COAD clinical matrix for 38 samples retrieved from GDC using TCGAbiolinks*

Description

TCGA-COAD clinical matrix for 38 samples retrieved from GDC using TCGAbiolinks

Usage

```
clinical
```

Format

A matrix: 38 samples (rows) and variables (columns) patient, sample, gender and sample_type

---

`cor_dnam_target_gene` *Evaluate correlation of DNA methylation region and target gene expression*

Description

This function evaluate the correlation of the DNA methylation and target gene expression using spearman rank correlation test. Note that genes with RNA expression equal to 0 for all samples will not be evaluated.

Usage

```
cor_dnam_target_gene(
  pair.dnam.target,
  dnam,
  exp,
  filter.results = TRUE,
  min.cor.pval = 0.05,
  min.cor.estimate = 0,
  cores = 1
)
```

Arguments

`pair.dnam.target`	A dataframe with the following columns: regionID (DNA methylation) and target (target gene)
`dnam`	DNA methylation matrix or SummarizedExperiment object with regions/cpgs in rows and samples in columns are samples. Samples should be in the same order as gene expression matrix (exp).
`exp`	Gene expression matrix or SummarizedExperiment object (rows are genes, columns are samples) log2-normalized (log2(exp + 1)). Samples should be in the same order as the DNA methylation matrix.
`filter.results`	Filter results using min.cor.pval and min.cor.estimate thresholds
`min.cor.pval`	P-value threshold filter (default: 0.05)
`min.cor.estimate`	Correlation estimate threshold filter (default: not applied)
`cores`	Number of CPU cores to be used. Default 1.

Value

A data frame with the following information: regionID, target gene, correlation pvalue and estimate between DNA methylation and target gene expression, FDR corrected p-values.

Examples

```
dnam <- t(matrix(sort(c(runif(20))), ncol = 1))
rownames(dnam) <- c("chr3:203727581-203728580")
colnames(dnam) <- paste0("Samples",1:20)
exp <- dnam
rownames(exp) <- c("ENSG00000232886")
colnames(exp) <- paste0("Samples",1:20)

pair.dnam.target <- data.frame(
   "regionID" =  c("chr3:203727581-203728580"),
   "target" = "ENSG00000232886"
)

# Correlated DNAm and gene expression, display only significant associations
results.cor.pos <- cor_dnam_target_gene(
   pair.dnam.target = pair.dnam.target,
   dnam = dnam,
   exp = exp,
   filter.results = TRUE,
   min.cor.pval = 0.05,
   min.cor.estimate = 0.0
)
```

---

cor_tf_target_gene *Evaluate correlation of TF expression and target gene expression*

Description

This function evaluate the correlation of a TF and target gene expression using spearman rank correlation test. Note that genes with RNA expression equal to 0 for all samples will not be evaluated.

Usage

```
cor_tf_target_gene(
  pair.tf.target,
  exp,
  tf.activity.es,
  cores = 1,
  verbose = FALSE
)
```

Arguments

pair.tf.target	A dataframe with the following columns: TF and target (target gene)
exp	Gene expression matrix or SummarizedExperiment object (rows are genes, columns are samples) log2-normalized (log2(exp + 1)). Samples should be in the same order as the tf.activity.es matrix
tf.activity.es	A matrix with normalized enrichment scores for each TF across all samples to be used in linear models instead of TF gene expression. See get_tf_ES.
cores	Number of CPU cores to be used. Default 1.
verbose	Show messages ?

Value

A data frame with the following information: TF, target gene, correlation p-value and estimate between TF and target gene expression, FDR corrected p-values.

Examples

```
exp <- t(matrix(sort(c(runif(40))), ncol = 2))
rownames(exp) <- c("ENSG00000232886","ENSG00000232889")
colnames(exp) <- paste0("Samples",1:20)

pair.tf.target <- data.frame(
   "TF" = "ENSG00000232889",
   "target" = "ENSG00000232886"
)

# Correlated TF and gene expression
results.cor.pos <- cor_tf_target_gene(
   pair.tf.target = pair.tf.target,
   exp = exp,
)
# Correlated TF and gene expression
results.cor.pos <- cor_tf_target_gene(
   pair.tf.target = pair.tf.target,
   exp = exp,
   tf.activity.es = exp
)
```

---

create_triplet_distance_based

Map DNAm to target genes using distance approaches, and TF to the DNAm region using JASPAR2020 TFBS.

Description

This function wraps two other functions `get_region_target_gene` and `get_tf_in_region` from the package. This function will map a region to a target gene using three methods (mapping to the closest gene, mapping to any gene within a given window of distance, or mapping to a fixed number of nearby genes upstream or downstream). To find TFs binding to the region, JASPAR2020 is used.

Usage

```
create_triplet_distance_based(
  region,
  genome = c("hg38", "hg19"),
 target.method = c("genes.promoter.overlap", "window", "nearby.genes", "closest.gene"),
  target.window.size = 500 * 10^3,
  target.num.flanking.genes = 5,
  target.promoter.upstream.dist.tss = 2000,
  target.promoter.downstream.dist.tss = 2000,
  target.rm.promoter.regions.from.distal.linking = TRUE,
  motif.search.window.size = 0,
  motif.search.p.cutoff = 1e-08,
```

```
  TF.peaks.gr = NULL,
  max.distance.region.target = 10^6,
  cores = 1
)
```

Arguments

region	A Granges or a named vector with regions (i.e "chr21:100002-1004000")
genome	Human genome reference "hg38" or "hg19"
target.method	How genes are mapped to regions: regions overlapping gene promoter ("genes.promoter.overlap"); genes within a window around the region ("window"); or fixed number of nearby genes upstream and downstream from the region
target.window.size	When method = "window", number of base pairs to extend the region (+- window.size/2). Default is 500kbp (or +/- 250kbp, i.e. 250k bp from start or end of the region)
target.num.flanking.genes	Number of flanking genes upstream and downstream to search. For example, if target.num.flanking.genes = 5, it will return the 5 genes upstream and 5 genes downstream
target.promoter.upstream.dist.tss	Number of base pairs (bp) upstream of TSS to consider as promoter regions. Defaults to 2000 bp.
target.promoter.downstream.dist.tss	Number of base pairs (bp) downstream of TSS to consider as promoter regions. Defaults to 2000 bp.
target.rm.promoter.regions.from.distal.linking	When performing distal linking with method = "windows" or method = "nearby.genes", or "closest.gene.tss", if set to TRUE (default), probes in promoter regions will be removed from the input.
motif.search.window.size	Integer value to extend the regions. For example, a value of 50 will extend 25 bp upstream and 25 downstream the region. Default is no increase
motif.search.p.cutoff	motifmatchr pvalue cut-off. Default 1e-8.
TF.peaks.gr	A granges with TF peaks to be overlaped with input region Metadata column expected "id" with TF name. Default NULL. Note that Remap catalog can be used as shown in the examples.
max.distance.region.target	Max distance between region and target gene. Default 1Mbp.
cores	Number of CPU cores to be used. Default 1.

Value

A data frame with TF, target and RegionID information.

Examples

```
regions.names <- c("chr3:189631389-189632889","chr4:43162098-43163498")
triplet <- create_triplet_distance_based(
```

```
    region = regions.names,
    motif.search.window.size = 500,
    target.method = "closest.gene"
)
```

`create_triplet_regulon_based` *Map TF and target genes using regulon databases or any user provided target-tf. Maps TF to the DNAm region with TFBS using JASPAR2020 TFBS.*

Description

This function wraps two other functions `get_region_target_gene` and `get_tf_in_region` from the package.

Usage

```
create_triplet_regulon_based(
  region,
  genome = c("hg38", "hg19"),
  regulons.min.confidence = "B",
  motif.search.window.size = 0,
  motif.search.p.cutoff = 1e-08,
  cores = 1,
  tf.target,
  TF.peaks.gr = NULL,
  max.distance.region.target = 10^6
)
```

Arguments

`region`	A Granges or a named vector with regions (i.e "chr21:100002-1004000")
`genome`	Human genome reference "hg38" or "hg19"
`regulons.min.confidence`	Minimun confidence score ("A", "B","C","D", "E") classifying regulons based on their quality from Human DoRothEA database dorothea_hs. The default minimun confidence score is "B".
`motif.search.window.size`	Integer value to extend the regions. For example, a value of 50 will extend 25 bp upstream and 25 downstream the region. Default is no increase
`motif.search.p.cutoff`	motifmatchr pvalue cut-off. Default 1e-8.
`cores`	Number of CPU cores to be used. Default 1.
`tf.target`	A dataframe with tf and target columns. If not provided, dorothea_hs will be used.
`TF.peaks.gr`	A granges with TF peaks to be overlaped with input region Metadata column expected "id" with TF name. Default NULL. Note that Remap catalog can be used as shown in the examples.
`max.distance.region.target`	Max distance between region and target gene. Default 1Mbp.

Value

A data frame with TF, target and RegionID information.

Examples

```
triplet <- create_triplet_regulon_based(
   region = c("chr1:69591-69592", "chr1:898803-898804"),
   motif.search.window.size = 50,
   regulons.min.confidence = "B",
     motif.search.p.cutoff = 0.05
)
```

---

dna.met.chr21

TCGA-COAD DNA methylation matrix (beta-values) for 38 samples (only chr21) retrieved from GDC using TCGAbiolinks

---

Description

TCGA-COAD DNA methylation matrix (beta-values) for 38 samples (only chr21) retrieved from GDC using TCGAbiolinks

Usage

```
dna.met.chr21
```

Format

A beta-value matrix with 38 samples, includes CpG IDs in the rows and TCGA sample identifiers in the columns

---

filter_dnam_by_quant_diff

Select regions with variations in DNA methylation levels above a threshold

---

Description

For each region, compares the mean DNA methylation (DNAm) levels in samples with high DNAm (Q4) vs. low DNAm (Q1) and requires the difference to be above a threshold.

Usage

```
filter_dnam_by_quant_diff(dnam, diff.mean.th = 0.2, cores = 1)
```

Arguments

dnam	DNA methylation matrix or SumarizedExperiment object
diff.mean.th	Threshold for difference in mean DNAm levels for samples in Q4 and Q1
cores	Number of CPU cores to be used in the analysis. Default: 1

Value

A subset of the original matrix only with the rows passing the filter threshold.

Examples

```
data("dna.met.chr21")
dna.met.chr21.filtered <- filter_dnam_by_quant_diff(
  dna.met.chr21
)
```

`filter_exp_by_quant_mean_FC`

Select genes with variations above a threshold

Description

For each gene, compares the mean gene expression levels in samples in high expression (Q4) vs. samples with low gene expression (Q1), and requires the fold change to be above a certain threshold.

Usage

```
filter_exp_by_quant_mean_FC(exp, fold.change = 1.5, cores = 1)
```

Arguments

`exp`	Gene expression matrix or SumarizedExperiment object
`fold.change`	Threshold for fold change of mean gene expression levels in samples with high (Q4) and low (Q1) gene expression levels. Defaults to 1.5.
`cores`	Number of CPU cores to be used in the analysis. Default: 1

Value

A subset of the original matrix only with the rows passing the filter threshold.

Examples

```
data("gene.exp.chr21.log2")
gene.exp.chr21.log2.filtered <- filter_exp_by_quant_mean_FC(
  gene.exp.chr21.log2
)
```

`filter_genes_zero_expression` *Remove genes with gene expression level equal to 0 in a substantial percentage of the samples*

Description

Remove genes with gene expression level equal to 0 in a substantial percentage of the samples

Usage

```
filter_genes_zero_expression(exp, max.samples.percentage = 0.25)
```

Arguments

`exp`	Gene expression matrix or SumarizedExperiment object
`max.samples.percentage`	Max percentage of samples with gene expression as 0, for genes to be selected. If max.samples.percentage 100, remove genes with 0 for 100% samples. If max.samples.percentage 25, remove genes with 0 for more than 25% of the samples.

Value

A subset of the original matrix only with the rows passing the filter threshold.

`gene.exp.chr21.log2` *TCGA-COAD gene expression matrix (log2 (FPKM-UQ + 1)) for 38 samples (only chromosome 21) retrieved from GDC using TCGAbiolinks*

Description

TCGA-COAD gene expression matrix (log2 (FPKM-UQ + 1)) for 38 samples (only chromosome 21) retrieved from GDC using TCGAbiolinks

Usage

```
gene.exp.chr21.log2
```

Format

A log2 (FPKM-UQ + 1) gene expression matrix with 38 samples, includes Ensembl IDs in the rows and TCGA sample identifiers in the columns

get_human_tfs *Access human TF from Lambert et al 2018*

Description

Access human TF from Lambert et al 2018 (PMID: 29425488)

Usage

```
get_human_tfs()
```

Value

A dataframe with Human TF

Examples

```
human.tfs <- get_human_tfs()
```

get_met_probes_info *Get HM450/EPIC manifest files from Sesame package*

Description

Returns a data frame with HM450/EPIC manifest information files from Sesame package

Usage

```
get_met_probes_info(genome = c("hg38", "hg19"), arrayType = c("450k", "EPIC"))
```

Arguments

genome	Human genome of reference hg38 or hg19
arrayType	"450k" or "EPIC" array

Examples

```
regions.names <- c("chr22:18267969-18268249","chr23:18267969-18268249")
regions.gr <- make_granges_from_names(regions.names)
make_names_from_granges(regions.gr)
```

get_promoter_avg *Summarize promoter DNA methylation beta values by mean.*

Description

First, identify gene promoter regions (default +-2Kkb around TSS). Then, for each promoter region calculate the mean DNA methylation of probes overlapping the region.

Usage

```
get_promoter_avg(
  dnam,
  genome,
  arrayType,
  cores = 1,
  upstream.dist.tss = 2000,
  downstream.dist.tss = 2000,
  verbose = FALSE
)
```

Arguments

dnam	A DNA methylation matrix or a SummarizedExperiment object
genome	Human genome of reference hg38 or hg19
arrayType	DNA methylation array type (450k or EPIC)
cores	A integer number to use multiple cores. Default 1 core.
upstream.dist.tss	Number of base pairs (bp) upstream of TSS to consider as promoter regions
downstream.dist.tss	Number of base pairs (bp) downstream of TSS to consider as promoter regions
verbose	A logical argument indicating if messages output should be provided.

Value

A RangedSummarizedExperiment with promoter region and mean beta-values of CpGs within it. Metadata will provide the promoter gene region and gene informations.

Examples

```
## Not run:
  data("dna.met.chr21")
  promoter.avg <- get_promoter_avg(
     dnam = dna.met.chr21,
     genome = "hg19",
     arrayType = "450k"
)

## End(Not run)
```

get_region_target_gene *Obtain target genes of input regions based on distance*

Description

To map an input region to genes there are three options: 1) map region to closest gene tss 2) map region to all genes within a window around the region (default window.size = 500kbp (i.e. +/- 250kbp from start or end of the region)). 3) map region to a fixed number of nearby genes (upstream/downstream)

Usage

```
get_region_target_gene(
  regions.gr,
  genome = c("hg38", "hg19"),
 method = c("genes.promoter.overlap", "window", "nearby.genes", "closest.gene.tss"),
  promoter.upstream.dist.tss = 2000,
  promoter.downstream.dist.tss = 2000,
  window.size = 500 * 10^3,
  num.flanking.genes = 5,
  rm.promoter.regions.from.distal.linking = TRUE
)
```

Arguments

regions.gr	A Genomic Ranges object (GRanges) or a SummarizedExperiment object (rowRanges will be used)
genome	Human genome of reference "hg38" or "hg19"
method	How genes are mapped to regions: region overlapping gene promoter ("genes.promoter.overlap"); or genes within a window around the region ("window"); or a fixed number genes upstream and downstream of the region ("nearby.genes"); or closest gene tss to the region ("closest.gene.tss")
promoter.upstream.dist.tss	Number of base pairs (bp) upstream of TSS to consider as promoter regions. Defaults to 2000 bp.
promoter.downstream.dist.tss	Number of base pairs (bp) downstream of TSS to consider as promoter regions. Defaults to 2000 bp.
window.size	When method = "window", number of base pairs to extend the region (+- window.size/2). Default is 500kbp (or +/- 250kbp, i.e. 250k bp from start or end of the region)
num.flanking.genes	When method = "nearby.genes", set the number of flanking genes upstream and downstream to search.Defaults to 5. For example, if num.flanking.genes = 5, it will return the 5 genes upstream and 5 genes downstream of the given region.
rm.promoter.regions.from.distal.linking	When performing distal linking with method = "windows", "nearby.genes" or "closest.gene.tss", if set to TRUE (default), probes in promoter regions will be removed from the input.

Details

For the analysis of probes in promoter regions (promoter analysis), we recommend setting `method = "genes.promoter.overlap"`.

For the analysis of probes in distal regions (distal analysis), we recommend setting either `method = "window"` or `method = "nearby.genes"`.

Note that because `method = "window"` or `method = "nearby.genes"` are mainly used for analyzing distal probes, by default `rm.promoter.regions.from.distal.linking = TRUE` to remove probes in promoter regions.

Value

A data frame with the following information: regionID, Target symbol, Target ensembl ID

Examples

```
library(GenomicRanges)
library(dplyr)

# Create example region
regions.gr <- data.frame(
       chrom = c("chr22", "chr22", "chr22", "chr22", "chr22"),
       start = c("39377790", "50987294", "19746156", "42470063", "43817258"),
       end   = c("39377930", "50987527", "19746368", "42470223", "43817384"),
       stringsAsFactors = FALSE)  %>%
     makeGRangesFromDataFrame

 # map to closest gene tss
 region.genes.promoter.overlaps <- get_region_target_gene(
                      regions.gr = regions.gr,
                      genome = "hg19",
                      method = "genes.promoter.overlap"
 )

 # map to all gene within region +- 250kbp
 region.window.genes <- get_region_target_gene(
                      regions.gr = regions.gr,
                      genome = "hg19",
                      method = "window",
                      window.size = 500 * 10^3
 )

 # map regions to n upstream and n downstream genes
 region.nearby.genes <- get_region_target_gene(
                      regions.gr = regions.gr,
                      genome = "hg19",
                      method = "nearby.genes",
                      num.flanking.genes = 5
 )
```

`get_residuals` *Get residuals from regression model*

Description

Compute studentized residuals from fitting linear regression models to expression values in a data matrix

Usage

```
get_residuals(data, metadata.samples = NULL, metadata.genes = NULL, cores = 1)
```

Arguments

data	A matrix or SummarizedExperiment object with samples as columns and features (gene, probes) as rows. Note that expression values should typically be log2(expx + 1) transformed before fitting linear regression models.
metadata.samples	A data frame with samples as rows and columns the covariates. No NA values are allowed, otherwise residual of the corresponding sample will be NA.
metadata.genes	A data frame with genes (covariates) as rows and samples as columns. For each evaluated gene, each column (e.g. CNA) that corresponds to the same gene will be set as a single covariate variable. This can be used to correct copy number alterations for each gene.
cores	Number of CPU cores to be used. Defaults to 1.

Details

When only `metadata.samples` are provided, this function computes residuals for expression values in a data matrix by fitting model

`features ~ Sample_covariate1 + Sample_covariate2 ... + Sample_covariateN` where `N` is the index of the columns in the metadata provided, `features` are (typically log transformed) expression values.

When the user additionally provide `metadata.genes`, that is, gene metadata (e.g. gene_covariate = copy number variations/alterations) residuals are computed by fitting the following model:

`features ~ Sample_covariate1 + Sample_covariate2 ... + Sample_covariateN + gene_covariate`

Value

A residuals matrix with samples as columns and features (gene, probes) as rows

Examples

```
data("gene.exp.chr21.log2")

data("clinical")
metadata <- clinical[,c( "gender", "sample_type")]

cnv <- matrix(
   sample(x = c(-2,-1,0,1,2),
   size = ncol(gene.exp.chr21.log2) * nrow(gene.exp.chr21.log2),replace = TRUE),
   nrow = nrow(gene.exp.chr21.log2),
   ncol = ncol(gene.exp.chr21.log2)
)
rownames(cnv) <- rownames(gene.exp.chr21.log2)
colnames(cnv) <- colnames(gene.exp.chr21.log2)
```

```
gene.exp.residuals <- get_residuals(
   data = gene.exp.chr21.log2[1:3,],
   metadata.samples = metadata,
   metadata.genes = cnv
)
gene.exp.residuals <- get_residuals(
   data = gene.exp.chr21.log2[1:3,],
   metadata.samples = metadata,
   metadata.genes = cnv[1:2,]
)
gene.exp.residuals <- get_residuals(
   data = gene.exp.chr21.log2[1:3,],
   metadata.samples = metadata
)
```

`get_tf_ES` *Calculate enrichment scores for each TF across all samples using dorothea and viper.*

Description

Calculate enrichment scores for each TF across all samples using dorothea and viper.

Usage

```
get_tf_ES(exp, min.confidence = "B", regulons)
```

Arguments

`exp`	Gene expression matrix with gene expression counts, row as ENSG gene IDS and column as samples
`min.confidence`	Minimun confidence score ("A", "B","C","D", "E") classifying regulons based on their quality from Human DoRothEA database. The default minimun confidence score is "B"
`regulons`	DoRothEA regulons in table format. Same as run_viper. If not specified Bioconductor (human) dorothea regulons besed on GTEx will be. used dorothea_hs.

Value

A matrix of normalized enrichment scores for each TF across all samples

Examples

```
gene.exp.chr21.log2 <- get(data("gene.exp.chr21.log2"))
tf_es <- get_tf_ES(gene.exp.chr21.log2)
```

`get_tf_in_region` *Get human TFs for regions by either scanning it with motifmatchr using JASPAR 2020 database or overlapping with TF chip-seq from user input*

Description

Given a genomic region, this function maps TF in regions using two methods: 1) using motifmatchr nd JASPAR2020 to scan the region for 554 human transcription factors binding sites. There is also an option (argument `window.size`) to extend the scanning region before performing the search, which by default is 0 (do not extend). 2) Using user input TF chip-seq to check for overlaps between region and TF peaks.

Usage

```
get_tf_in_region(
  region,
  window.size = 0,
  genome = c("hg19", "hg38"),
  p.cutoff = 1e-08,
  cores = 1,
  TF.peaks.gr = NULL,
  verbose = FALSE
)
```

Arguments

`region`	A vector of region names or GRanges object with the DNA methylation regions to be scanned for the motifs
`window.size`	Integer value to extend the regions. For example, a value of 50 will extend 25 bp upstream and 25 bp downstream the region. The default is not to increase the scanned region.
`genome`	Human genome of reference "hg38" or "hg19".
`p.cutoff`	motifmatchr p.cutoff. Default 1e-8.
`cores`	Number of CPU cores to be used. Default 1.
`TF.peaks.gr`	A granges with TF peaks to be overlaped with input region Metadata column expected "id" with TF name. Default NULL. Note that Remap catalog can be used as shown in the examples.
`verbose`	A logical argument indicating if messages output should be provided.

Value

A data frame with the following information: regionID, TF symbol, TF ensembl ID

Examples

```
regions.names <- c("chr3:189631389-189632889","chr4:43162098-43163498")
region.tf <- get_tf_in_region(
               region = regions.names,
               genome = "hg38"
```

```
 )

## Not run:
   library(ReMapEnrich)
   demo.dir <- "~/ReMapEnrich_demo"
   dir.create(demo.dir, showWarnings = FALSE, recursive = TRUE)
   # Use the function DowloadRemapCatalog
   remapCatalog2018hg38 <- downloadRemapCatalog(demo.dir, assembly = "hg38")
   # Load the ReMap catalogue and convert it to Genomic Ranges
   remapCatalog <- bedToGranges(remapCatalog2018hg38)
   regions.names <- c("chr3:189631389-189632889","chr4:43162098-43163498")
   region.tf.remap <- get_tf_in_region(
                   region = regions.names,
                   genome = "hg38",
                   TF.peaks.gr = remapCatalog
   )

## End(Not run)
```

`interaction_model` *Fits linear models with interaction to triplet data (Target, TF, DNAm), where DNAm is a binary variable (samples in Q1 or Q4)*

Description

Evaluates regulatory potential of DNA methylation (DNAm) on gene expression, by fitting robust linear model or zero inflated negative binomial model to triplet data. These models consist of terms to model direct effect of DNAm on target gene expression, direct effect of TF on gene expression, as well as an interaction term that evaluates the synergistic effect of DNAm and TF on gene expression.

Usage

```
interaction_model(
  triplet,
  dnam,
  exp,
  cores = 1,
  tf.activity.es = NULL,
  sig.threshold = 0.05,
  fdr = TRUE,
  filter.correlated.tf.exp.dnam = TRUE,
  filter.triplet.by.sig.term = TRUE,
  stage.wise.analysis = FALSE,
  verbose = FALSE
)
```

Arguments

`triplet`	Data frame with columns for DNA methylation region (regionID), TF (TF), and target gene (target)
`dnam`	DNA methylation matrix or SummarizedExperiment object (columns: samples in the same order as `exp` matrix, rows: regions/probes)

exp	A matrix or SummarizedExperiment object object (columns: samples in the same order as dnam, rows: genes represented by ensembl IDs (e.g. ENSG00000239415))
cores	Number of CPU cores to be used. Default 1.
tf.activity.es	A matrix with normalized enrichment scores for each TF across all samples to be used in linear models instead of TF gene expression. See get_tf_ES.
sig.threshold	Threshold to filter significant triplets. Select if interaction.pval < 0.05 or pval.dnam <0.05 or pval.tf < 0.05 in binary model
fdr	Uses fdr when using sig.threshold. Select if interaction.fdr < 0.05 or fdr.dnam <0.05 or fdr.tf < 0.05 in binary model
filter.correlated.tf.exp.dnam	If wilcoxon test of TF expression Q1 and Q4 is significant (pvalue < 0.05), triplet will be removed.
filter.triplet.by.sig.term	Filter significant triplets ? Select if interaction.pval < 0.05 or pval.dnam <0.05 or pval.tf < 0.05 in binary model
stage.wise.analysis	A boolean indicating if stagewise analysis should be performed to correct for multiple comparisons. If set to FALSE FDR analysis is performed.
verbose	A logical argument indicating if messages output should be provided.

Details

This function fits the linear model

`log2(RNA target) ~ log2(TF) + DNAm + log2(TF) * DNAm`

to triplet data as follow:

Model by considering DNAm as a binary variable - we defined a binary group for DNA methylation values (high = 1, low = 0). That is, samples with the highest DNAm levels (top 25 percent) has high = 1, samples with lowest DNAm levels (bottom 25 percent) has high = 0. Note that in this implementation, only samples with DNAm values in the first and last quartiles are considered.

In these models, the term `log2(TF)` evaluates direct effect of TF on target gene expression, DNAm evaluates direct effect of DNAm on target gene expression, and `log2(TF)*DNAm` evaluates synergistic effect of DNAm and TF, that is, if TF regulatory activity is modified by DNAm.

There are two implementations of these models, depending on whether there are an excessive amount (i.e. more than 25 percent) of samples with zero counts in RNAseq data:

- When percent of zeros in RNAseq data is less than 25 percent, robust linear models are implemented using `rlm` function from MASS package. This gives outlier gene expression values reduced weight. We used `"psi.bisqure"` option in function `rlm` (bisquare weighting, https://stats.idre.ucla.edu/r/dae/robust-regression/).
- When percent of zeros in RNAseq data is more than 25 percent, zero inflated negative binomial models are implemented using `zeroinfl` function from `pscl` package. This assumes there are two processes that generated zeros (1) one where the counts are always zero (2) another where the count follows a negative binomial distribution.

To account for confounding effects from covariate variables, first use the `get_residuals` function to obtain RNA or DNAm residual values which have covariate effects removed, then fit interaction model. Note that no log2 transformation is needed when `interaction_model` is applied to residuals data.

Note that only triplets with TF expression not significantly different in high vs. low methylation groups will be evaluated (Wilcoxon test, p > 0.05).

Value

A dataframe with Region,TF,target,TF_symbo,target_symbol,estimates and P-values, after fitting robust linear models or zero-inflated negative binomial models (see Details above).

Model considering DNAm values as a binary variable generates quant_pval_metGrp, quant_pval_rna.tf, quant_pval_metGrp.rna.tf, quant_estimates_metGrp, quant_estimates_rna.tf, quant_estimates_metGrp.rn

Model.interaction indicates which model (robust linear model or zero inflated model) was used to fit Model 1, and Model.quantile indicates which model(robust linear model or zero inflated model) was used to fit Model 2.

Examples

```
library(dplyr)
dnam <- runif(20,min = 0,max = 1) %>%
  matrix(ncol = 1) %>%  t
rownames(dnam) <- c("chr3:203727581-203728580")
colnames(dnam) <- paste0("Samples",1:20)

exp.target <-  runif(20,min = 0,max = 10) %>%
  matrix(ncol = 1) %>%  t
rownames(exp.target) <- c("ENSG00000232886")
colnames(exp.target) <- paste0("Samples",1:20)

exp.tf <- runif(20,min = 0,max = 10) %>%
  matrix(ncol = 1) %>%  t
rownames(exp.tf) <- c("ENSG00000232888")
colnames(exp.tf) <- paste0("Samples",1:20)

exp <- rbind(exp.tf, exp.target)

triplet <- data.frame(
   "regionID" =  c("chr3:203727581-203728580"),
   "target" = "ENSG00000232886",
   "TF" = "ENSG00000232888"
)
results <- interaction_model(triplet, dnam, exp)
```

---

make_dnam_se *Transform DNA methylation array into a summarized Experiment object*

Description

Transform DNA methylation array into a summarized Experiment object

Usage

```
make_dnam_se(
  dnam,
  genome = c("hg38", "hg19"),
  arrayType = c("450k", "EPIC"),
  betaToM = FALSE,
  verbose = FALSE
)
```

Arguments

dnam	DNA methylation matrix with beta-values or m-values as data, row as cpgs "cg07946458" or regions ("chr1:232:245") and column as samples
genome	Human genome of reference: hg38 or hg19
arrayType	DNA methylation array type (450k or EPIC)
betaToM	indicates if converting methylation beta values to mvalues
verbose	A logical argument indicating if messages output should be provided.

Value

A summarized Experiment object with DNA methylation probes mapped to genomic regions

Examples

```
library(dplyr)
dnam <- runif(20, min = 0,max = 1) %>% sort %>%
  matrix(ncol = 1) %>%  t
rownames(dnam) <- c("chr3:203727581-203728580")
colnames(dnam) <- paste0("Samples",1:20)
 se <- make_dnam_se(dnam)
```

---

make_exp_se *Transform gene expression matrix into a Summarized Experiment object*

Description

Transform gene expression matrix into a Summarized Experiment object

Usage

```
make_exp_se(exp, genome = c("hg38", "hg19"), verbose = FALSE)
```

Arguments

exp	Gene expression matrix with gene expression counts, row as ENSG gene IDS and column as samples
genome	Human genome of reference: hg38 or hg19
verbose	A logical argument indicating if messages output should be provided.

Value

A summarized Experiment object

Examples

```
gene.exp.chr21.log2 <- get(data("gene.exp.chr21.log2"))
gene.exp.chr21.log2.se <- make_exp_se(gene.exp.chr21.log2)
```

make_granges_from_names *Create a Granges object from a genmic region string*

Description

Given a region name such as chr22:18267969-18268249, we will create a Granges object

Usage

```
make_granges_from_names(names)
```

Arguments

names	A region name as "chr22:18267969-18268249" or a vector of region names.

Examples

```
regions.names <- c("chr22:18267969-18268249","chr23:18267969-18268249")
regions.gr <- make_granges_from_names(regions.names)
```

make_names_from_granges *Create region name from Granges*

Description

Given a GRanges returns region name such as chr22:18267969-18268249

Usage

```
make_names_from_granges(region)
```

Arguments

region	A GenomicRanges object

Examples

```
regions.names <- c("chr22:18267969-18268249","chr23:18267969-18268249")
regions.gr <- make_granges_from_names(regions.names)
make_names_from_granges(regions.gr)
```

MethReg *MethReg: functional annotation of DMRs identified in epigenome-wide association studies*

Description

To provide functional annotations for differentially methylated regions (DMRs) and differentially methylated CpG sites (DMS), MethReg performs integrative analyses using matched DNA methylation and gene expression along with Transcription Factor Binding Sites (TFBS) data. MethReg evaluates, prioritizes and annotates DNA methylation regions (or sites) with high regulatory potential that works synergistically with TFs to regulate target gene expressions, without any additional ChIP-seq data.

plot_interaction_model *Plot interaction model results*

Description

Create several plots to show interaction data TF expression with target gene interaction using a linear model

$$log2(RNAtarget) = log2(TF) + DNAm + log2(TF) * DNAm$$

To consider covariates, RNA can also be the residuals.

$$log2(RNAtargetresiduals) = log2(TFresidual) + DNAm + log2(TFresidual) * DNAm$$

Usage

```
plot_interaction_model(
  triplet.results,
  dnam,
  exp,
  metadata,
  tf.activity.es = NULL,
  tf.dnam.classifier.pval.thld = 0.001,
  label.dnam = "beta-value",
  label.exp = "expression",
  genome = "hg38"
)
```

Arguments

triplet.results	Output from function interaction_model with Region ID, TF (column name: TF), and target gene (column name: target), p-values and estimates of interaction
dnam	DNA methylation matrix or SummarizedExperiment object (columns: samples same order as met, rows: regions/probes)

exp	gene expression matrix or a SummarizedExperiment object (columns: samples same order as met, rows: genes)
metadata	A data frame with samples as rownames and one columns that will be used to color the samples
tf.activity.es	A matrix with normalized enrichment scores for each TF across all samples to be used in linear models instead of TF gene expression.
tf.dnam.classifier.pval.thld	P-value threshold to consider a linear model significant of not. Default 0.001. This will be used to classify the TF role and DNAm effect.
label.dnam	Used for label text. Option "beta-value" and "residuals"
label.exp	Used for label text. Option "expression" and "residuals"
genome	Genome of reference to be added to the plot as text

Value

A ggplot object, includes a table with results from fitting interaction model, and the the following scatter plots: 1) TF vs DNAm, 2) Target vs DNAm, 3) Target vs TF, 4) Target vs TF for samples in Q1 and Q4 for DNA methylation, 5) Target vs DNAm for samples in Q1 and Q4 for the TF

Examples

```
library(dplyr)
dnam <- runif(20, min = 0,max = 1) %>% sort %>%
  matrix(ncol = 1) %>%  t
rownames(dnam) <- c("chr3:203727581-203728580")
colnames(dnam) <- paste0("Samples",1:20)

exp.target <-  runif(20,min = 0,max = 10) %>% sort %>%
  matrix(ncol = 1) %>%  t
rownames(exp.target) <- c("ENSG00000232886")
colnames(exp.target) <- paste0("Samples",1:20)

exp.tf <- runif(20,min = 0,max = 10) %>%
  matrix(ncol = 1) %>%  t
rownames(exp.tf) <- c("ENSG00000101412")
colnames(exp.tf) <- paste0("Samples",1:20)

exp <- rbind(exp.tf, exp.target)

triplet <- data.frame(
   "regionID" =  c("chr3:203727581-203728580"),
   "target" = "ENSG00000232886",
   "TF" = "ENSG00000101412"
)

results <- interaction_model(
  triplet = triplet,
  dnam = dnam,
  exp = exp,
  fdr = FALSE,
  filter.correlated.tf.exp.dna = FALSE
)
plots <- plot_interaction_model(
    triplet.results = results,
```

```
    dnam = dnam,
    exp = exp
)
```

`plot_stratified_model` *Plot stratified model results*

Description

Create several plots to show interaction data TF expression with target gene interaction using a linear model

$$log2(RNAtarget)\ log2(TF)$$

to samples with highest DNAm values (top 25 percent) and lowest DNAm values (bottom 25 percent), separately.

Usage

```
plot_stratified_model(
  triplet.results,
  dnam,
  exp,
  metadata,
  label.dnam = "beta-value",
  label.exp = "expression",
  tf.activity.es = NULL
)
```

Arguments

`triplet.results`	Output from function stratified_model with Region ID, TF (column name: TF), and target gene (column name: target), p-values and estimates of interaction
`dnam`	DNA methylation matrix or SummarizedExperiment object (columns: samples same order as met, rows: regions/probes)
`exp`	A gene expression matrix or SummarizedExperiment object (columns: samples same order as met, rows: genes)
`metadata`	A data frame with samples as rownames and one columns that will be used to color the samples
`label.dnam`	Used for label text. Option "beta-value" and "residuals"
`label.exp`	Used for label text. Option "expression" and "residuals"
`tf.activity.es`	A matrix with normalized enrichment scores for each TF across all samples to be used in linear models instead of TF gene expression.

Value

A ggplot object, includes a table with results from fitting stratified model, and the following scatter plots: 1) TF vs DNAm, 2) Target vs DNAm, 3) Target vs TF, 4) Target vs TF for samples in Q1 and Q4 for DNA methylation, 5) Target vs DNAm for samples in Q1 and Q4 for the TF

Examples

```
library(dplyr)
dnam <- runif(20,min = 0,max = 1) %>%
  matrix(ncol = 1) %>%  t
rownames(dnam) <- c("chr3:203727581-203728580")
colnames(dnam) <- paste0("Samples",1:20)

exp.target <-  runif(20,min = 0,max = 10) %>%
  matrix(ncol = 1) %>%  t
rownames(exp.target) <- c("ENSG00000232886")
colnames(exp.target) <- paste0("Samples",1:20)

exp.tf <- runif(20,min = 0,max = 10) %>%
  matrix(ncol = 1) %>%  t
rownames(exp.tf) <- c("ENSG00000232888")
colnames(exp.tf) <- paste0("Samples",1:20)

exp <- rbind(exp.tf, exp.target)

triplet <- data.frame(
   "regionID" =  c("chr3:203727581-203728580"),
   "target" = "ENSG00000232886",
   "TF" = "ENSG00000232888"
)

results <- stratified_model(triplet = triplet,dnam = dnam, exp = exp)
plots <- plot_stratified_model(
    triplet.results = results,
    dnam = dnam,
    exp = exp
)
```

`stratified_model` *Fits linear models to triplet data (Target, TF, DNAm) for samples with high DNAm or low DNAm separately, and annotates TF (activator/repressor) and DNam effect over TF activity (attenuate, enhance).*

Description

Should be used after fitting `interaction_model`, and only for triplet data with significant `TF*DNAm` interaction. This analysis examines in more details on how TF activities differ in samples with high DNAm or low DNAm values.

Usage

```
stratified_model(
  triplet,
  dnam,
  exp,
  cores = 1,
  tf.activity.es = NULL,
  tf.dnam.classifier.pval.thld = 0.001
)
```

Arguments

triplet	Data frame with columns for DNA methylation region (regionID), TF (TF), and target gene (target)
dnam	DNA methylation matrix or SummarizedExperiment (columns: samples in the same order as exp matrix, rows: regions/probes)
exp	A matrix or SummarizedExperiment (columns: samples in the same order as dnam matrix, rows: genes represented by ensembl IDs (e.g. ENSG00000239415))
cores	Number of CPU cores to be used. Default 1.
tf.activity.es	A matrix with normalized enrichment scores for each TF across all samples to be used in linear models instead of TF gene expression.
tf.dnam.classifier.pval.thld	P-value threshold to consider a linear model significant of not. Default 0.001. This will be used to classify the TF role and DNAm effect.

Details

This function fits linear model `log2(RNA target) = log2(TF)`

to samples with highest DNAm values (top 25 percent) or lowest DNAm values (bottom 25 percent), separately.

There are two implementations of these models, depending on whether there are an excessive amount (i.e. more than 25 percent) of samples with zero counts in RNAseq data:

- When percent of zeros in RNAseq data is less than 25 percent, robust linear models are implemented using `rlm` function from `MASS` package. This gives outlier gene expression values reduced weight. We used `"psi.bisqure"` option in function `rlm` (bisquare weighting, https://stats.idre.ucla.edu/r/dae/robust-regression/).
- When percent of zeros in RNAseq data is more than 25 percent, zero inflated negative binomial models are implemented using `zeroinfl` function from `pscl` package. This assumes there are two processes that generated zeros (1) one where the counts are always zero (2) another where the count follows a negative binomial distribution.

To account for confounding effects from covariate variables, first use the `get_residuals` function to obtain RNA residual values which have covariate effects removed, then fit interaction model. Note that no log2 transformation is needed when `interaction_model` is applied to residuals data.

This function also provides annotations for TFs. A TF is annotated as `activator` if increasing amount of TF (higher TF gene expression) corresponds to increased target gene expression. A TF is annotated as `repressor` if increasing amount of TF (higher TF gene expression) corresponds to decrease in target gene expression. A TF is annotated as `dual` if in the Q1 methylation group increasing amount of TF (higher TF gene expression) corresponds to increase in target gene expression, while in Q4 methylation group increasing amount of TF (higher TF gene expression) corresponds to decrease in target gene expression (or the same but changing Q1 and Q4 in the previous sentence).

In addition, a region/CpG is annotated as `enhancing` if more TF regulation on gene transcription is observed in samples with high DNAm. That is, DNA methylation enhances TF regulation on target gene expression. On the other hand, a region/CpG is annotated as `attenuating` if more TF regulation on gene transcription is observed in samples with low DNAm. That is, DNA methylation reduces TF regulation on target gene expression.

Value

A dataframe with `Region`, `TF`, `target`, `TF_symbol` `target_symbol`, results for fitting linear models to samples with low methylation (`DNAmlow_pval_rna.tf`, `DNAmlow_estimate_rna.tf`), or samples with high methylation (`DNAmhigh_pval_rna.tf`, `DNAmhigh_pval_rna.tf.1`), annotations for TF (`class.TF`) and (`class.TF.DNAm`).

Examples

```
library(dplyr)
dnam <- runif (20,min = 0,max = 1) %>%
  matrix(ncol = 1) %>%  t
rownames(dnam) <- c("chr3:203727581-203728580")
colnames(dnam) <- paste0("Samples",1:20)

exp.target <-  runif (20,min = 0,max = 10) %>%
  matrix(ncol = 1) %>%  t
rownames(exp.target) <- c("ENSG00000232886")
colnames(exp.target) <- paste0("Samples",1:20)

exp.tf <- runif (20,min = 0,max = 10) %>%
  matrix(ncol = 1) %>%  t
rownames(exp.tf) <- c("ENSG00000232888")
colnames(exp.tf) <- paste0("Samples",1:20)

exp <- rbind(exp.tf, exp.target)

triplet <- data.frame(
   "regionID" =  c("chr3:203727581-203728580"),
   "target" = "ENSG00000232886",
   "TF" = "ENSG00000232888"
)

results <- stratified_model(
  triplet = triplet,
  dnam = dnam,
  exp = exp
)
```

Index