Package ‘coMethDMR’

May 9, 2024

Title Accurate identification of co-methylated and differentially methylated regions in epigenome-wide association studies

Version 1.8.0

Description coMethDMR identifies genomic regions associated with continuous phenotypes by optimally leverages covariations among CpGs within predefined genomic regions. Instead of testing all CpGs within a genomic region, coMethDMR carries out an additional step that selects co-methylated sub-regions first without using any outcome information. Next, coMethDMR tests association between methylation within the sub-region and continuous phenotype using a random coefficient mixed effects model, which models both variations between CpG sites within the region and differential methylation simultaneously.

Depends R (>= 4.1)

License GPL-3

Encoding UTF-8

LazyData false

RoxygenNote 7.1.2

Imports AnnotationHub, BiocParallel, bumphunter, ExperimentHub, GenomicRanges, IRanges, lmerTest, methods, stats, utils

Suggests BiocStyle, corrplot, knitr, rmarkdown, testthat, IlluminaHumanMethylation450kanno.ilmn12.hg19, IlluminaHumanMethylationEPICanno.ilm10b4.hg19

biocViews DNAMethylation, Epigenetics, MethylationArray, DifferentialMethylation, GenomeWideAssociation

VignetteBuilder knitr

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

URL https://github.com/TransBioInfoLab/coMethDMR

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

git_branch RELEASE_3_19

git_last_commit 0588fe1
Contents

AnnotateResults ................................................. 3
betaMatrix_ex1 .................................................. 4
betaMatrix_ex2 .................................................. 4
betaMatrix_ex3 .................................................. 5
betaMatrix_ex4 .................................................. 5
betasChr22_df ................................................. 6
CloseBySingleRegion ........................................... 6
CoMethAllRegions .............................................. 7
coMethDMR_setup ............................................. 9
CoMethSingleRegion .......................................... 10
CpGsInfoAllRegions .......................................... 11
CpGsInfoOneRegion .......................................... 13
CreateOutputDF ............................................... 14
CreateParallelWorkers ....................................... 15
CreateRdrop .................................................. 16
FindComethylatedRegions ................................... 17
GetCpGsInRegion ............................................ 18
GetResiduals .................................................. 19
ImportSesameData ............................................. 20
lmmTest ...................................................... 21
lmmTestAllRegions .......................................... 23
MarkComethylatedCpGs .................................... 25
MarkMissing ................................................... 27
NameRegion ..................................................... 28
OrderCpGsByLocation ....................................... 29
pheno_df ....................................................... 30
RegionsToRanges ............................................. 30
SplitCpGDFbyRegion ......................................... 31
WriteCloseByAllRegions ................................... 32

Index ............................................................. 34
Description

Given a data frame with regions in the genome, add gene symbols, UCSC reference gene accession, UCSC reference gene group and relation to CpG island.

Usage

AnnotateResults(lmmRes_df, arrayType = c("450k", "EPIC"), nCores_int = 1L, ...)

Arguments

lmmRes_df  A data frame returned by lmmTestAllRegions. This data frame must contain the following columns:

• chrom: the chromosome the region is on, e.g. “chr22”
• start: the region start point
• end: the region end point

arrayType  Type of array: 450k or EPIC

nCores_int  Number of computing cores to be used when executing code in parallel. Defaults to 1 (serial computing).

...  Dots for additional arguments passed to the cluster constructor. See CreateParallelWorkers for more information.

Details

The region types include "NSHORE", "NSHELF", "SSHORE", "SSHELF", "TSS1500", "TSS200", "UTR5", "EXON1", "GENEBODY", "UTR3", and "ISLAND".

Value

A data frame with

• the location of the genomic region’s chromosome (chrom), start (start), and end (end);
• UCSC annotation information (UCSC_RefGene_Group, UCSC_RefGene_Accession, and UCSC_RefGene_Name); and
• a list of all of the probes in that region (probes).

Examples

lmmResults_df <- data.frame(
  chrom = c("chr22", "chr22", "chr22", "chr22", "chr22"),
  start = c("39377790", "50987294", "19746156", "42470063", "43817258"),
  end = c("39377930", "50987527", "19746368", "42470223", "43817384"),
  regionType = c("TSS1500", "EXON1", "ISLAND", "TSS200", "ISLAND"),
  ...
betaMatrix_ex2

Alzheimer’s Prefrontal Cortex (PFC) Methylation Data

Description
Subset of an Alzheimer’s Disease methylation data set, with beta values for measured CpGs methylation levels.

Usage
data("betaMatrix_ex2")

Format
A data frame containing beta values for 4 CpGs in one CpG islands for 110 subjects. Each column is a CpG, each row is a sample.

Source
GEO accession: GSE59685

betaMatrix_ex1

Alzheimer’s Prefrontal Cortex (PFC) Methylation Data

Description
Subset of an Alzheimer’s Disease methylation data set, with beta values for measured CpGs methylation levels.

Usage
data("betaMatrix_ex1")

Format
A data frame containing beta values for 4 CpGs in one CpG islands for 110 subjects. Each column is a CpG, each row is a sample.
Source

GEO accession: GSE59685

---

**betaMatrix_ex3**  
*Alzheimer’s Prefrontal Cortex (PFC) Methylation Data*

Description

Subset of an Alzheimer’s Disease methylation data set, with beta values for measured CpGs methylation levels.

Usage

`data("betaMatrix_ex3")`

Format

A data frame containing beta values for 6 CpGs in one CpG islands for 110 subjects. Each column is a CpG, each row is a sample.

Source

GEO accession: GSE59685

---

**betaMatrix_ex4**  
*Alzheimer’s Prefrontal Cortex (PFC) Methylation Data*

Description

Subset of an Alzheimer’s Disease methylation data set, with beta values for measured CpGs methylation levels.

Usage

`data("betaMatrix_ex4")`

Format

A data frame containing beta values for 52 CpGs in one CpG islands for 110 subjects. Each column is a CpG, each row is a sample.

Source

GEO accession: GSE59685
betasChr22_df  

*Prefrontal Cortex (PFC) Methylation Data from Alzheimer's Disease subjects*

**Description**

Subset of an Alzheimer's methylation dataset, with beta values for CpGs.

**Usage**

```r
data("betasChr22_df")
```

**Format**

A data frame containing beta values for 8552 CpGs in Chr22 for a subset of 20 subjects.

**Source**

GEO accession: GSE59685

---

CloseBySingleRegion  

*Extract clusters of CpGs located closely in a genomic region.*

**Description**

Extract clusters of CpGs located closely in a genomic region.

**Usage**

```r
CloseBySingleRegion(
  CpGs_char, 
  genome = c("hg19", "hg38"),
  arrayType = c("450k", "EPIC"),
  manifest_gr = NULL,
  maxGap = 200,
  minCpGs = 3
)
```

**Arguments**

- **CpGs_char**: A list of CpG IDs
- **genome**: Human genome of reference hg19 or hg38
- **arrayType**: Type of array, 450k or EPIC
CoMethAllRegions

A GRanges object with the genome manifest (as returned by ExperimentHub or by ImportSesameData). This function by default ignores this argument in favour of the genome and arrayType arguments.

maxGap

an integer, genomic locations within maxGap from each other are placed into the same cluster

minCpGs

an integer, minimum number of CpGs for the resulting CpG cluster

Details

Note that this function depends only on CpG locations, and not on any methylation data. The algorithm is based on the clusterMaker function in the bumphunter R package. Each cluster is essentially a group of CpG locations such that two consecutive locations in the cluster are separated by less than maxGap.

Value

a list, each item in the list is a character vector of CpG IDs located closely (i.e. in the same cluster)

Examples

CpGs_char <- c(
  "cg02505293", "cg03618257", "cg04421269", "cg17885402", "cg19890033",
  "cg20566587", "cg27505880"
)

cluster_ls <- CloseBySingleRegion(
  CpGs_char,
  genome = "hg19",
  arrayType = "450k",
  maxGap = 100,
  minCpGs = 3
)

CoMethAllRegions

Extract contiguous co-methylated genomic regions from a list of pre-defined genomic regions

Description

Extract contiguous co-methylated genomic regions from a list of pre-defined genomic regions

Usage

CoMethAllRegions(
  dnam,
  betaToM = FALSE,
  method = c("pearson", "spearman"),

rDropThresh_num = 0.4, 
minCpGs = 3, 
genome = c("hg19", "hg38"), 
arrayType = c("450k", "EPIC"), 
CpGs_ls, 
file = NULL, 
returnAllCpGs = FALSE, 
output = c("CpGs", "dataframe"), 
nCores_int = 1L, 
... 
)

Arguments

dnam matrix (or data frame) of beta values, with row names = CpG IDs, column names = sample IDs. This is typically genome-wide methylation beta values.
betaToM indicates if converting methylation beta values to mvalues
method method for computing correlation, can be "spearman" or "pearson"
rDropThresh_num threshold for min correlation between a cpg with sum of the rest of the CpGs
minCpGs minimum number of CpGs to be considered a "region". Only regions with more than minCpGs will be returned.
genome Human genome of reference hg19 or hg38
arrayType Type of array, can be "450k" or "EPIC"
CpGs_ls list where each item is a character vector of CpGs IDs. This should be CpG probes located closely on the array.
file an RDS file with clusters of CpG locations (i.e. CpGs located closely to each other on the genome). This file can be generated by the WriteCloseByAllRegions function.
returnAllCpGs When there is not a contiguous comethylated region in the inputting pre-defined region, returnAllCpGs = TRUE indicates outputting all the CpGs in the input regions (regardless of statistical significance), while returnAllCpGs = FALSE indicates not returning any CpGs not contained in comethylated clusters. Defaults to FALSE, and we provide this option for debugging purposes only.
output a character vector of CpGs or a dataframe of CpGs along with rDrop info
nCores_int Number of computing cores to be used when executing code in parallel. Defaults to 1 (serial computing).
... Dots for additional arguments passed to the cluster constructor. See CreateParallelWorkers for more information.

Details

There are two ways to input genomic regions for this function: (1) use CpGs_ls argument, or (2) use file argument. Examples of these files are in /inst/extdata/ folder of the package.
Value

When output = "dataframe" is selected, returns a list of data frames, each with CpG (CpG name), Chr (chromosome number), MAPINFO (genomic position), r_drop (correlation between the CpG with rest of the CpGs), keep (indicator for co-methylated CpG), keep_contiguous (index for contiguous co-methylated subregions).

When output = "CpGs" is selected, returns a list, each item is a list of CpGs in the contiguous co-methylated subregion.

Examples

data(betasChr22_df)

CpGisland_ls <- readRDS(
  system.file(
    "extdata",
    "CpGislandsChr22_ex.rds",
    package = 'coMethDMR',
    mustWork = TRUE
  )
)

coMeth_ls <- CoMethAllRegions(
  dnam = betasChr22_df,
  betaToM = TRUE,
  method = "pearson",
  CpGs_ls = CpGisland_ls,
  arrayType = "450k",
  returnAllCpGs = FALSE,
  output = "CpGs"
)

---

coMethDMR_setup  Cache sesameData at Package Load

Description

Check if the user has both the HM540 and EPIC manifests in their cache. The contents of the cache are checked via a call to the ExperimentHub function. If not all data components are available in the cache for these two platforms, we query the necessary data to save them to the cache for later use.

Arguments

libname     path to package library
pkgname     package name
CoMethSingleRegion

Details

arguments are unused

---

CoMethSingleRegion

Wrapper function to find contiguous and comethylated sub-regions within a pre-defined genomic region

---

Description

Wrapper function to find contiguous and comethylated sub-regions within a pre-defined genomic region

Usage

CoMethSingleRegion(
  CpGs_char,
  dnam,
  betaToM = TRUE,
  rDropThresh_num = 0.4,
  method = c("pearson", "spearman"),
  minCpGs = 3,
  genome = c("hg19", "hg38"),
  arrayType = c("450k", "EPIC"),
  manifest_gr = NULL,
  returnAllCpGs = FALSE
)

Arguments

- **CpGs_char**: vector of CpGs in the inputting pre-defined genomic region.
- **dnam**: matrix (or data frame) of beta values, with row names = CpG ids, column names = sample ids. This should include the CpGs in CpGs_char, as well as additional CpGs.
- **betaToM**: indicates if converting methylation beta values mvalues
- **rDropThresh_num**: threshold for min correlation between a cpg with sum of the rest of the CpGs
- **method**: method for computing correlation, can be "pearson" or "spearman"
- **minCpGs**: minimum number of CpGs to be considered a "region". Only regions with more than minCpGs will be returned.
- **genome**: Human genome of reference hg19 or hg38
- **arrayType**: Type of array, can be "450k" or "EPIC"
- **manifest_gr**: A GRanges object with the genome manifest (as returned by ExperimentHub or by ImportSesameData). This function by default ignores this argument in favour of the genome and arrayType arguments.
- **returnAllCpGs**: When there is not a contiguous comethylated region in the inputting pre-defined region, returnAllCpGs = 1 indicates outputting all the CpGs in the input region, while returnAllCpGs = 0 indicates not returning any CpG.
Value

A list with two components:

- **Contiguous_Regions**: a data frame with CpG (CpG ID), Chr (chromosome number), MAPINFO (genomic position), r_drop (correlation between the CpG with rest of the CpGs), keep (indicator for co-methylated CpG), keep_contiguous (index for contiguous co-methylated subregion).

- **CpGs_subregions**: lists of CpGs in each contiguous co-methylated subregion.

Examples

```r
data(betasChr22_df)
CpGsChr22_char <- c("cg02953382", "cg12419862", "cg24565820", "cg04234412", "cg04824771", "cg09033563", "cg10150615", "cg18538332", "cg20007245", "cg23131131", "cg25703541")
CoMethSingleRegion(
  CpGs_char = CpGsChr22_char,
  dnam = betasChr22_df
)

data(betaMatrix_ex3)
CpGsEx3_char <- c("cg14221598", "cg02433884", "cg07372974", "cg13419809", "cg26856676", "cg25246745")
CoMethSingleRegion(
  CpGs_char = CpGsEx3_char,
  dnam = t(betaMatrix_ex3),
  returnAllCpGs = TRUE
)
```

Description

Test associations of individual CpGs in multiple genomic regions with a continuous phenotype.

Usage

```r
CpGsInfoAllRegions(
  AllRegionNames_char,
  allRegions_gr = NULL,
  betas_df,
```
CpGsInfoAllRegions

```r
pheno_df,
contPheno_char,
covariates_char,
genome = c("hg19", "hg38"),
arrayType = c("450k", "EPIC")
)
```

**Arguments**

- `AllRegionNames_char` vector of character strings with location info for all the genomic regions. Each region should be specified in this format: "chrxx:xxxxxx-xxxxxx"
- `allRegions_gr` An object of class `GRanges` with location information for the regions. If this argument is NULL, then the regions in `AllRegionNames_char` are used. If this argument is not NULL, then `region_gr` will overwrite any supplied ranges in `AllRegionNames_char`.
- `betas_df` data frame of beta values for all genomic regions, with row names = CpG IDs and column names = sample IDs
- `pheno_df` a data frame with phenotype and covariate variables, with variable "Sample" for sample IDs.
- `contPheno_char` character string of the continuous phenotype to be tested against methylation values
- `covariates_char` character vector of covariate variables names
- `genome` human genome of reference hg19 (default) or hg38
- `arrayType` Type of array, can be "450k" or "EPIC"

**Value**

a data frame with locations of the genomic region (Region), CpG ID (cpg), chromosome (chr), position (pos), results for testing association of methylation in individual CpGs with the continuous phenotype (slopeEstimate, slopePval), UCSC_RefGene_Name, UCSC_RefGene_Accession, and UCSC_RefGene_Group

**Examples**

```r
data(betasChr22_df)
data(pheno_df)
AllRegionNames_char <- c("chr22:18267969-18268249", "chr22:18531243-18531447")
CpGsInfoAllRegions(
  AllRegionNames_char,
  betas_df = betasChr22_df,
  pheno_df = pheno_df,
  contPheno_char = "stage",
)```
CpGsInfoOneRegion  

```r
covariates_char = c("age.brain", "sex")
```

---

## Description

Test associations of individual CpGs in a genomic region with a continuous phenotype

## Usage

```r
CpGsInfoOneRegion(
  regionName_char,
  region_gr = NULL,
  betas_df,
  pheno_df,
  contPheno_char,
  covariates_char = NULL,
  genome = c("hg19", "hg38"),
  arrayType = c("450k", "EPIC"),
  manifest_gr = NULL
)
```

## Arguments

- `regionName_char` character string of location information for a genomic region, specified in the format of "chrxx:xxxxxx-xxxxxx"
- `region_gr` An object of class `GRanges` with location information for one region. If this argument is NULL, then the region in `regionName_char` is used.
- `betas_df` data frame of beta values with row names = CpG IDs, column names = sample IDs
- `pheno_df` a data frame with phenotype and covariate variables, with variable "Sample" for sample IDs.
- `contPheno_char` character string of the continuous phenotype to be tested against methylation values
- `covariates_char` character vector of covariate variables names
- `genome` human genome of reference hg19 (default) or hg38
- `arrayType` Type of array, can be "450k" or "EPIC"
- `manifest_gr` A GRanges object with the genome manifest (as returned by `ExperimentHub` or by `ImportSesameData`). This function by default ignores this argument in favour of the genome and `arrayType` arguments.
CreateOutputDF

Details

This function implements linear models that test association between methylation values in a genomic region with a continuous phenotype. Note that methylation M values are used as regression outcomes in these models. The model for each CpG is:

\[
\text{methylation M value} \sim \text{contPheno_char} + \text{covariates_char}
\]

Value

A data frame with location of the genomic region (Region), CpG ID (cpg), chromosome (chr), position (pos), results for testing association of methylation in individual CpGs with continuous phenotype (slopeEstimate, slopePval) and annotations for the region.

Examples

```r
data(betasChr22_df)
data(pheno_df)
myRegion_gr <- RegionsToRanges("chr22:18267969-18268249")

CpGsInfoOneRegion(
  region_gr = myRegion_gr,
  betas_df = betasChr22_df,
  pheno_df = pheno_df,
  contPheno_char = "stage",
  covariates_char = c("age.brain", "sex"),
  arrayType = "450k"
)
```

CreateOutputDF

Create Output Dataframe

Description

Create Output Dataframe

Usage

```r
CreateOutputDF(
  keepCpGs_df,
  keepContiguousCpGs_df,
  CpGsOrdered_df,
  returnAllCpGs = FALSE
)
```
CreateParallelWorkers

Arguments

- `keepCpGs_df` a data frame with `CpG = CpG name`, `keep = indicator for co-methylated CpGs`, and `r_drop = correlation between the CpG with rest of the CpGs`
- `keepContiguousCpGs_df` a data frame with `ProbeID = CpG name` and `Subregion = contiguous comethylated subregion number`
- `CpGsOrdered_df` a data frame of CpG location with `chr = chromosome number`, `pos = genomic position`, and `cpg = CpG name`
- `returnAllCpGs` indicates if outputting all the CpGs in the region when there is not a contiguous comethylated region or only the CpGs in the contiguous comethylated regions

Value

a data frame with `CpG = CpG name`, `Chr = chromosome number`, `MAPINFO = genomic position`, `r_drop = correlation between the CpG with rest of the CpGs`, `keep = indicator for co-methylated CpG`, and `keep_contiguous = contiguous comethylated subregion number`

Examples

data(betasChr22_df)
CpGsChr22_char <- c(
  "cg02953382", "cg12419862", "cg24565820", "cg04234412", "cg04824771",
  "cg09833563", "cg10150615", "cg18538332", "cg20007245", "cg23131131",
  "cg25703541"
)

CpGsOrdered_df <- OrderCpGsByLocation(
  CpGsChr22_char, arrayType="450k", output = "dataframe"
)

betaCluster_mat <- t(betasChr22_df[CpGsOrdered_df$cpg, ])
keepCpGs_df <- MarkComethylatedCpGs(betaCluster_mat = betaCluster_mat)
keepContiguousCpGs_df <- FindComethylatedRegions(CpGs_df = keepCpGs_df)
CreateOutputDF(keepCpGs_df, keepContiguousCpGs_df, CpGsOrdered_df)

---

CreateParallelWorkers

Create a Parallel Computing Cluster

Description

This function is an operating-system agnostic wrapper for the SnowParam and MulticoreParam constructor functions.

Usage

CreateParallelWorkers(nCores, ...)
Arguments

nCores: The number of computing cores to make available for coMethDMR computation

... Additional arguments passed to the cluster constructors.

Details

This function checks the operating system and then creates a cluster of workers using the `SnowParam` function for Windows machines and the `MulticoreParam` function for non-Windows machines.

Value

A parameter class for use in parallel evaluation

Examples

```r
workers_cl <- CreateParallelWorkers(nCores = 4)
```

<table>
<thead>
<tr>
<th>CreateRdrop</th>
<th>Computes leave-one-out correlations (rDrop) for each CpG</th>
</tr>
</thead>
</table>

Description

Computes leave-one-out correlations (rDrop) for each CpG

Usage

```r
CreateRdrop(data, method = c("pearson", "spearman"), use = "complete.obs")
```

Arguments

- data: a dataframe with rownames = sample IDs, column names = CpG IDs.
- method: method for computing correlation, can be "pearson" or "spearman", and is passed to the cor function.
- use: method for handling missing values when calculating the correlation. Defaults to "complete.obs" because the option "pairwise.complete.obs" only works for Pearson correlation.

Details

An outlier CpG in a genomic region will typically have low correlation with the rest of the CpGs in a genomic region. On the other hand, in a cluster of co-methylated CpGs, we expect each CpG to have high correlation with the rest of the CpGs. The r.drop statistic is used to identify these co-methylated CpGs here.
FindComethylatedRegions

Value

A data frame with the following columns:

- **CpG**: CpG ID
- **r_drop**: The correlation between each CpG with the sum of the rest of the CpGs

Examples

```r
data(betaMatrix_ex1)
CreateRdrop(data = betaMatrix_ex1, method = "pearson")
```

FindComethylatedRegions

*Find Contiguous Co-Methylated Regions*

Description

Find contiguous comethylated regions based on output file from function MarkComethylatedCpGs

Usage

```r
FindComethylatedRegions(CpGs_df, minCpGs_int = 3)
```

Arguments

- **CpGs_df**: an output dataframe from function MarkComethylatedCpGs, with variables: CpG, keep, ind, r_drop. See details in documentation for `MarkComethylatedCpGs`
- **minCpGs_int**: an integer indicating the minimum number of CpGs for output genomic regions

Value

A data frame with variables `ProbeID` and `Subregion` (index for each output contiguous comethylated region)

Examples

```r
data(betaMatrix_ex4)
CpGs_df <- MarkComethylatedCpGs(betaCluster_mat = betaMatrix_ex4)
FindComethylatedRegions(CpGs_df)
```
GetCpGsInRegion

Extract probe IDs for CpGs located in a genomic region

Description

Extract probe IDs for CpGs located in a genomic region

Usage

GetCpGsInRegion(
  regionName_char,
  region_gr = NULL,
  genome = c("hg19", "hg38"),
  arrayType = c("450k", "EPIC"),
  manifest_gr = NULL,
  ignoreStrand = TRUE
)

Arguments

regionName_char  character string with location information for one region in the format "chrxx:xxxxxx-xxxxxx"
region_gr        An object of class GRanges with location information for one region. If this argument is NULL, then the region in regionName_char is used.
genome            human genome of reference hg19 (default) or hg38
arrayType         Type of array, 450k or EPIC
manifest_gr       A GRanges object with the genome manifest (as returned by ExperimentHub or by ImportSesameData). This function by default ignores this argument in favour of the genome and arrayType arguments.
ignoreStrand      Whether strand can be ignored, default is TRUE

Value

vector of CpG probe IDs mapped to the genomic region

Examples

myRegion_gr <- RegionsToRanges("chr22:18267969-18268249")

GetCpGsInRegion(
  region_gr = myRegion_gr,
  genome = "hg19",
  arrayType = "450k",
  ignoreStrand = TRUE
)
GetResiduals

Description
Remove covariate effects from methylation values by fitting probe-specific linear models

Usage
GetResiduals(
dnam,
betaToM = TRUE,
epsilon = 1e-08,
pheno_df,
covariates_char,
nCores_int = 1L,
...
)

Arguments
dnam data frame or matrix of methylation values with row names = CpG IDs and column names = sample IDs. This is often the genome-wide array data.
betaToM indicates if methylation beta values (ranging from [0, 1]) should be converted to M values (ranging from (-Inf, Inf)). Note that if beta values are the input to dnam, then betaToM should be set to TRUE, otherwise FALSE.
epsilon When transforming beta values to M values, what should be done to values exactly equal to 0 or 1? The M value transformation would yield -Inf or Inf which causes issues in the statistical model. We thus replace all values exactly equal to 0 with 0 + epsilon, and we replace all values exactly equal to 1 with 1 - epsilon. Defaults to epsilon = 1e-08.
pheno_df a data frame with phenotype and covariates, with variable Sample indicating sample IDs.
covariates_char character vector for names of the covariate variables
nCores_int Number of computing cores to be used when executing code in parallel. Defaults to 1 (serial computing).
... Dots for additional arguments passed to the cluster constructor. See CreateParallelWorkers for more information.

Details
This function fits an ordinary linear model predicting methylation values for each probe from the specified covariates. This process will be useful in scenarios where methylation values in a region or at an individual probe are known a priori to have differential methylation independent of the disease or condition of interest.
Value

output a matrix of residual values in the same dimension as dnam

Examples

data(betasChr22_df)
data(pheno_df)

GetResiduals(
  dnam = betasChr22_df[1:10, 1:10],
  betaToM = TRUE,
  pheno_df = pheno_df,
  covariates_char = c("age.brain", "sex", "slide")
)

---

ImportSesameData

Import Illumina manifests (sesameData versions)

Description

Load either the HM540 and EPIC manifests into working memory

Usage

ImportSesameData(manifest_char)

Arguments

manifest_char Which manifest should be loaded? Currently, this package has been tested to work with 450k and EPIC arrays for HG19 and HG38.

Details

This function assumes that the .onLoad() function has executed properly and (therefore) that the necessary data sets are in the cache.

Examples

hm450k_gr <- ImportSesameData("HM450.hg19.manifest")
head(hm450k_gr)
Fit mixed model to methylation values in one genomic region

**Description**

Fit mixed model to methylation values in one genomic region

**Usage**

```r
lmmTest(
  betaOne_df,
  pheno_df,
  contPheno_char,
  covariates_char,
  modelType = c("randCoef", "simple"),
  genome = c("hg19", "hg38"),
  arrayType = c("450k", "EPIC"),
  manifest_gr = NULL,
  ignoreStrand = TRUE,
  outLogFile = NULL
)
```

**Arguments**

- `betaOne_df` matrix of beta values for one genomic region, with row names = CpG IDs and column names = sample IDs
- `pheno_df` a data frame with phenotype and covariates, with variable `Sample` indicating sample IDs.
- `contPheno_char` character string of the main effect (a continuous phenotype) to be tested for association with methylation values in the region
- `covariates_char` character vector for names of the covariate variables
- `modelType` type of mixed model: can be `randCoef` for random coefficient mixed model or `simple` for simple linear mixed model.
- `genome` Human genome of reference: `hg19` or `hg38`
- `arrayType` Type of array: "450k" or "EPIC"
- `manifest_gr` A GRanges object with the genome manifest (as returned by `ExperimentHub` or by `ImportSesameData`). This function by default ignores this argument in favour of the genome and arrayType arguments.
- `ignoreStrand` Whether strand can be ignored, default is `TRUE`
- `outLogFile` Name of log file for messages of mixed model analysis
Details

This function implements a mixed model to test association between methylation M values in a genomic region with a continuous phenotype. In our simulation studies, we found both models shown below are conservative, so p-values are estimated from normal distributions instead of Student’s t distributions.

When modelType = "randCoef", the model is:

\[ M \sim \text{contPheno\_char} + \text{covariates\_char} + (1|\text{Sample}) + (\text{contPheno\_char}|\text{CpG}). \]

The last term specifies random intercept and slope for each CpG. When modelType = "simple", the model is

\[ M \sim \text{contPheno\_char} + \text{covariates\_char} + (1|\text{Sample}). \]

Value

A dataframe with one row for association result of one region and the following columns: Estimate, StdErr, and pvalue showing the association of methylation values in the genomic region tested with the continuous phenotype supplied in contPheno_char.

Examples

data(betasChr22_df)

CpGsChr22_char <- c(
  "cg02953382", "cg12419862", "cg24565820", "cg04234412", "cg04824771",
  "cg09033563", "cg10150615", "cg18538332", "cg20007245", "cg23131131",
  "cg25703541"
)

coMethCpGs <- CoMethSingleRegion(CpGsChr22_char, betasChr22_df)

# test only the first co-methylated region
coMethBeta_df <- betasChr22_df[coMethCpGs$CpGsSubregions[[1]], ]

data(pheno_df)

res <- lmmTest(
  betaOne_df = coMethBeta_df, pheno_df, contPheno_char = "stage", covariates_char = c("age.brain", "sex"), modelType = "randCoef", arrayType = "450k", ignoreStrand = TRUE)
**Description**

Fit mixed model to test association between a continuous phenotype and methylation values in a list of genomic regions.

**Usage**

```r
lmmTestAllRegions(
  betas,
  region_ls,
  pheno_df,
  contPheno_char,
  covariates_char,
  modelType = c("randCoef", "simple"),
  genome = c("hg19", "hg38"),
  arrayType = c("450k", "EPIC"),
  ignoreStrand = TRUE,
  outFile = NULL,
  outLogFile = NULL,
  nCores_int = 1L,
  ...
)
```

**Arguments**

- `betas`: data frame or matrix of beta values for all genomic regions, with row names = CpG IDs and column names = sample IDs. This is often the genome-wide array data.
- `region_ls`: a list of genomic regions; each item is a vector of CpG IDs within a genomic region. The co-methylated regions can be obtained by function `CoMethAllRegions`.
- `pheno_df`: a data frame with phenotype and covariates, with variable `Sample` indicating sample IDs.
- `contPheno_char`: character string of the main effect (a continuous phenotype) to be tested for association with methylation values in each region.
- `covariates_char`: character vector for names of the covariate variables.
- `modelType`: type of mixed model; can be `randCoef` for random coefficient mixed model or `simple` for simple linear mixed model.
- `genome`: Human genome of reference: `hg19` or `hg38`.
- `arrayType`: Type of array: "450k" or "EPIC".
- `ignoreStrand`: Whether strand can be ignored, default is TRUE.


ImTestAllRegions

outFile output .csv file with the results for the mixed model analysis
outLogFile log file for mixed models analysis messages
nCores_int Number of computing cores to be used when executing code in parallel. Defaults to 1 (serial computing).

Details

This function implements a mixed model to test association between methylation M values in a genomic region with a continuous phenotype. In our simulation studies, we found both models shown below are conservative, so p-values are estimated from normal distributions instead of Student’s t distributions.

When `modelType = "randCoef"`, the model is:

\[ M \sim \text{contPheno_char} + \text{covariates_char} + (1|\text{Sample}) + (\text{contPheno_char}|\text{CpG}). \]

The last term specifies random intercept and slope for each CpG. When `modelType = "simple"`, the model is

\[ M \sim \text{contPheno_char} + \text{covariates_char} + (1|\text{Sample}). \]

For the results of mixed models, note that if the mixed model failed to converge, p-value for mixed model is set to 1. Also, if the mixed model is singular, at least one of the estimated variance components for intercepts or slopes random effects is 0, because there isn’t enough variability in the data to estimate the random effects. In this case, the mixed model reduces to a fixed effects model. The p-values for these regions are still valid.

Value

If `outFile` is NULL, this function returns a data frame as described below. If `outFile` is specified, this function writes a data frame in .csv format with the following information to the disk: location of the genomic region (chrom, start, end), number of CpGs (nCpGs), Estimate, Standard error (StdErr) of the test statistic, p-value and False Discovery Rate (FDR) for association between methylation values in each genomic region with phenotype (pValue).

If `outLogFile` is specified, this function also writes a .txt file that includes messages for mixed model fitting to the disk.

Examples

```r
data(betasChr22_df)
data(pheno_df)

CpGisland_ls <- readRDS(
  system.file(
    "extdata",
    "CpGislandsChr22_ex.rds",
    package = 'coMethDMR',
    mustWork = TRUE
  )
)
```
MarkComethylatedCpGs

Mark CpGs in contiguous and co-methylated region

Description

Mark CpGs in contiguous and co-methylated region

Usage

MarkComethylatedCpGs(
  betaCluster_mat,
  betaToM = TRUE,
  epsilon = 1e-08,
  rDropThresh_num = 0.4,
  method = c("pearson", "spearman"),
  use = "complete.obs"
)

Arguments

betaCluster_mat

matrix of beta values, with rownames = sample ids and column names = CpG ids. Note that the CpGs need to be ordered by their genomic positions, this can be accomplished by the OrderCpGbyLocation function.
betaToM indicates if beta values should be converted to M values before computing correlations. Defaults to TRUE.

epsilon When transforming beta values to M values, what should be done to values exactly equal to 0 or 1? The M value transformation would yield -Inf or Inf which causes issues in the statistical model. We thus replace all values exactly equal to 0 with 0 + epsilon, and we replace all values exactly equal to 1 with 1 - epsilon. Defaults to epsilon = 1e-08.

rDropThresh_num threshold for minimum correlation between a cpg with the rest of the CpGs. Defaults to 0.4.

method correlation method; can be "pearson" or "spearman"

use method for handling missing values when calculating the correlation. Defaults to "complete.obs" because the option "pairwise.complete.obs" only works for Pearson correlation.

Details

An outlier CpG in a genomic region will typically have low correlation with the rest of the CpGs in a genomic region. On the other hand, in a cluster of co-methylated CpGs, we expect each CpG to have high correlation with the rest of the CpGs. The \texttt{r.drop} statistic is used to identify these co-methylated CpGs here.

Value

A data frame with the following columns:

- \texttt{CpG} : CpG ID
- \texttt{keep} : The CpGs with \texttt{keep} = 1 belong to the contiguous and co-methylated region
- \texttt{ind} : Index for the CpGs
- \texttt{r_drop} : The correlation between each CpG with the sum of the rest of the CpGs

Examples

data(betaMatrix_ex1)

MarkComethylatedCpGs(
  betaCluster_mat = betaMatrix_ex1,
  betaToM = FALSE,
  method = "pearson"
)
MarkMissing

Return Column and Row Names of Samples and Probes under the Missingness Threshold

Description

Return Column and Row Names of Samples and Probes under the Missingness Threshold

Usage

MarkMissing(dnaM_df, sampMissing_p = 0.5, probeMissing_p = 0.25)

Arguments

dnaM_df A data frame of DNA methylation values. Samples are columns. Row names are probe IDs.
sampMissing_p The maximum proportion of missingness allowed in a sample. Defaults to 50%.
probeMissing_p The maximum proportion of missingness allowed in a probe. Defaults to 25%.

Details

Before calculating the missing proportion of samples, probes with missingness greater than the threshold are dropped first.

Value

A list of four entries:

- dropSamples: the column names of samples with more than sampMissing_p percent missing values
- keepSamples: the column names of samples with less than or equal to sampMissing_p percent missing values
- dropProbes: the row names of probes with more than probeMissing_p percent missing values
- keepProbes: the row names of probes with less than or equal to probeMissing_p percent missing values

Examples

```r
### Setup ###
values_num <- c(
  0.1, 0.1, 0.1, 0.1, 0.1,
  0.1, 0.1, 0.1, 0.1, NA,
  0.1, 0.1, 0.1, 0.1, NA,
  0.1, 0.1, 0.1, NA, NA,
  0.1, 0.1, NA, NA, NA,
```
values_mat <- matrix(values_num, nrow = 9, ncol = 5, byrow = TRUE)
rownames(values_mat) <- paste0("probe_0", 1:9)
colnames(values_mat) <- paste0("sample_0", 1:5)
values_df <- as.data.frame(values_mat)

### Simple Calculations ###
MarkMissing(values_df)
MarkMissing(values_df, probeMissing_p = 0.5)
MarkMissing(values_df, sampMissing_p = 0.25)

### Using the Output ###
mark_ls <- MarkMissing(values_df, probeMissing_p = 0.5)
valuesPurged_df <- values_df[mark_ls$keepProbes, mark_ls$keepSamples]
valuesPurged_df

---

NameRegion

Name a region with several Cpgs based on its genomic location

Description

Name a region with several Cpgs based on its genomic location

Usage

NameRegion(CpGsOrdered_df)

Arguments

CpGsOrdered_df dataframe with columns for Probe IDs as character (cpg), chromosome number as character (chr), and genomic location as integer (pos)

Value

gene location of the CpGs formatted as "chrxx:xxxxxx-xxxxxx"

Examples

# Consider four probe IDs:
CpGs_char <- c("cg04677227", "cg07146435", "cg11632906", "cg20214853")

# After querying these four probes against an EPIC array via the
# OrderCpGsByLocation() function, we get the following data frame:
CpGsOrdered_df <- data.frame(
OrderCpGsByLocation

Order CpGs by genomic location

Description
Order CpGs by genomic location

Usage
OrderCpGsByLocation(
  CpGs_char,
  genome = c("hg19", "hg38"),
  arrayType = c("450k", "EPIC"),
  manifest_gr = NULL,
  ignoreStrand = TRUE,
  output = c("vector", "dataframe")
)

Arguments
  
  Argument  Description
  
  CpGs_char  vector of CpGs
  genome  Human genome of reference: hg19 or hg38
  arrayType  Type of array: 450k or EPIC
  manifest_gr  A GRanges object with the genome manifest (as returned by ExperimentHub or by ImportSesameData). This function by default ignores this argument in favour of the genome and arrayType arguments.
  ignoreStrand  Whether strand can be ignored, default is TRUE
  output  vector of CpGs or dataframe with CpGs, CHR, MAPINFO

Value
vector of CpGs ordered by location or dataframe with CpGs ordered by location (cpg), chromosome (chr), position (pos)
Examples
CpGs_char <- c("cg04677227", "cg07146435", "cg11632906", "cg20214853")
OrderCpGsByLocation(
  CpGs_char,
  genome = "hg19",
  arrayType = "450k",
  ignoreStrand = TRUE,
  output = "dataframe"
)

---

**pheno_df**

*Example phenotype data file from Prefrontal Cortex (PFC) Methylation Data of Alzheimer’s Disease subjects*

---

**Description**

Subset of phenotype information for Alzheimer’s methylation dataset.

**Usage**

data("pheno_df")

**Format**

A data frame containing variables for Braak stage (stage), subject.id, Batch (slide), Sex, Sample, age of brain sample (age.brain)

**Source**

GEO accession: GSE59685

---

**RegionsToRanges**

*Convert genomic regions in a data frame to GRanges format*

---

**Description**

Convert genomic regions in a data frame to GRanges format

**Usage**

RegionsToRanges(regionName_char)

**Arguments**

regionName_char

a character vector of regions in the format "chrxx:xxxxx-xxxxxx"
SplitCpGDFbyRegion

Value

  genomic regions in GRanges format

Examples

  regions <- c("chr22:19709548-19709755", "chr2:241721922-241722113")
  RegionsToRanges(regions)

SplitCpGDFbyRegion  Split CpG dataframe by Subregion

Description

Split a dataframe of CpGs and comethylated subregions to a list of CpGs in each subregion

Usage

  SplitCpGDFbyRegion(
    CpGsSubregions_df, genome = c("hg19", "hg38"),
    arrayType = c("450k", "EPIC"),
    manifest_gr = NULL,
    returnAllCpGs = TRUE
  )

Arguments

  CpGsSubregions_df  data frame with CpG and subregion number
  genome  Human genome of reference: hg19 or hg38
  arrayType  Type of array: 450k or EPIC
  manifest_gr  A GRanges object with the genome manifest (as returned by ExperimentHub or by ImportSesameData). This function by default ignores this argument in favour of the genome and arrayType arguments.
  returnAllCpGs  indicates if outputting all the CpGs in the region when there is not a contiguous comethylated region or only the CpGs in the contiguous comethylated regions

Value

  a list of comethylated subregions CpGs for a pre-defined region
Examples

```r
data(betaMatrix_ex4)
CpGs_df <- MarkComethylatedCpGs(betaCluster_mat = betaMatrix_ex4)
CpGsSubregions_df <- FindComethylatedRegions(CpGs_df)

SplitCpGDFbyRegion(
  CpGsSubregions_df,
  genome = "hg19",
  arrayType = "450k"
)
```

WriteCloseByAllRegions

Extract clusters of CpG probes located closely

Description

Extract clusters of CpG probes located closely

Usage

```r
WriteCloseByAllRegions(
  fileName,
  regions,
  genome = c("hg19", "hg38"),
  arrayType = c("450k", "EPIC"),
  ignoreStrand = TRUE,
  maxGap = 200,
  minCpGs = 3,
  ...
)
```

Arguments

- `fileName`: Name of the RDS file where the output genomic regions will be saved.
- `regions`: GRanges of input genomic regions
- `genome`: Human genome of reference: hg19 or hg38
- `arrayType`: Type of array: "450k" or "EPIC"
- `ignoreStrand`: Whether strand can be ignored, default is TRUE
- `maxGap`: an integer, genomic locations within maxGap from each other are placed into the same cluster
- `minCpGs`: an integer, minimum number of CpGs for each resulting region
- `...`: Dots for internal arguments. Currently unused.
Details

For $\text{maxGap} = 200$ and $\text{minCpGs} = 3$, we have already calculated the clusters of CpGs. They are saved in folder `/inst/extdata/`.

Value

Nothing. Instead, file with the genomic regions containing CpGs located closely within each inputting pre-defined genomic region will be written to the disk

Examples

```r
regions <- GenomicRanges::GRanges(
  seqnames = c("chr4", "chr6", "chr16", "chr16", "chr22", "chr19"),
  ranges = c(
    "174202697-174203520", "28226203-28227482", "89572934-89574634",
    "67232460-67234167", "38244199-38245362", "39402823-39403373"
  )
)

# Uncomment out the example code below:
# WriteCloseByAllRegions(
#   regions = regions,
#   arrayType = "EPIC",
#   maxGap = 50,
#   minCpGs = 3,
#   fileName = "closeByRegions.rds"
# )
```
Index

* datasets
  - betaMatrix_ex1, 4
  - betaMatrix_ex2, 4
  - betaMatrix_ex3, 5
  - betaMatrix_ex4, 5
  - betasChr22_df, 6
  - pheno_df, 30

* internal
  - coMethDMR_setup, 9
  - CreateOutputDF, 14
  - SplitCpGDFbyRegion, 31
  - AnnotateResults, 3
  - betaMatrix_ex1, 4
  - betaMatrix_ex2, 4
  - betaMatrix_ex3, 5
  - betaMatrix_ex4, 5
  - betasChr22_df, 6
  - CloseBySingleRegion, 6
  - clusterMaker, 7
  - CoMethAllRegions, 7, 23
  - coMethDMR_setup, 9
  - CoMethSingleRegion, 10
  - cor, 16
  - CpGsInfoAllRegions, 11
  - CpGsInfoOneRegion, 13
  - CreateOutputDF, 14
  - CreateParallelWorkers, 3, 8, 15, 19, 24
  - CreateRdrop, 16
  - ExperimentHub, 7, 10, 13, 18, 21, 29, 31
  - FindComethylatedRegions, 17
  - GetCpGsInRegion, 18
  - GetResiduals, 19
  - GRanges, 12, 13, 18
  - ImportSesameData, 7, 10, 13, 18, 20, 21, 29, 31

  lmmTest, 21
  lmmTestAllRegions, 3, 23
  MarkComethylatedCpGs, 17, 25
  MarkMissing, 27
  MulticoreParam, 15
  NameRegion, 28
  OrderCpGsByLocation, 29
  pheno_df, 30
  query, 9
  RegionsToRanges, 30
  SnowParam, 15
  SplitCpGDFbyRegion, 31
  WriteCloseByAllRegions, 8, 32