Package 'coMethDMR'

October 10, 2019

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

Version 0.0.0.9001

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 (>= 3.5.0),
IlluminaHumanMethylation450kanno.ilmn12.hg19,
IlluminaHumanMethylationEPICanno.ilm10b2.hg19

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

Imports BiocParallel,
bumphunter,
GenomicRanges,
IRanges,
lmerTest,
stats,
utils

Suggests knitr,
testthat

biocViews DNAMethylation,
Epigenetics,
MethylationArray,
DifferentialMethylation,
GenomeWideAssociation

VignetteBuilder knitr

R topics documented:

`AnnotateResults` *Annotate* `coMethDMR` *Pipeline Results*

Description

Given a data frame with regions in the genome, add gene symbols, UCSC reference gene accession, and IDs of probes in the region.

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 Optionally, the data frame can also has `regionType`, which is a character string marking the type of genomic region tested. See details below.
`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"),
  stringsAsFactors = FALSE
)

AnnotateResults(
  lmmRes_df = lmmResults_df,
  arrayType = "450k"
)
```

---

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

```
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_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

```
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_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

```
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

```
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

```
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

```
CloseBySingleRegion(CpGs_char, arrayType = c("450k", "EPIC"),
  maxGap = 200, minCpGs = 3)
```

Arguments

CpGs_char	a list of CpG IDs
arrayType	Type of array, 450k or EPIC
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 clsuter 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, 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,
  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`	thershold for min correlation between a cpg with sum of the rest of the CpGs
`minCpGs`	mininum number of CpGs to be considered a "region". Only regions with more than `minCpGs` will be returned.
`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 inputing pre-defined region, `returnAllCpGs = 1` indicates outputting all the CpGs in the input regions, while `returnAllCpGs = 0` indicates not returning any CpG.
`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 (2) use `file` argument

examples of these files are at https://github.com/lissettegomez/coMethDMRdata

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 comethylated 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"
)
```

CoMethSingleRegion *Wrapper function to find contiguous and comethyalted sub-regions within a pre-defined genomic region*

Description

Wrapper function to find contiguous and comethyalted 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, arrayType = c("450k", "EPIC"), 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	thershold 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	mininum number of CpGs to be considered a "region". Only regions with more than minCpGs will be returned.
arrayType	Type of array, can be "450k" or "EPIC"
returnAllCpGs	When there is not a contiguous comethylated region in the inputing 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 comethylated subregion)
- CpGs_subregions : lists of CpGs in each contiguous co-methylated subregion

Examples

```
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
)
```

`CpGsInfoAllRegions` *Test associations of individual CpGs in multiple genomic regions with a continuous phenotype*

Description

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

Usage

```
CpGsInfoAllRegions(AllRegionNames_char, betas_df, pheno_df, contPheno_char,
  covariates_char, 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"
`betas_df`	data frame of beta values for all genomic regions, 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
`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), and results for testing association of methylation in individual CpGs with continuous phenotype (slopeEstimate, slopePval), UCSC_RefGene_Name, UCSC_RefGene_Accession, UCSC_RefGene_Group

Examples

```
data(betasChr22_df)
data(pheno_df)
AllRegionNames_char <- c("chr22:18267969-18268249", "chr22:18531243-18531447")

CpGsInfoAllRegions(
  AllRegionNames_char,
  betas_df = betasChr22_df,
  pheno_df, contPheno_char = "stage",
  covariates_char = c("age.brain", "sex")
)
```

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

Description

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

Usage

```
CpGsInfoOneRegion(regionName_char, betas_df, pheno_df, contPheno_char,
  covariates_char, arrayType = c("450k", "EPIC"))
```

Arguments

regionName_char	character string of location information for a genomic region, specified in the format of "chrxx:xxxxxx-xxxxxx"
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
arrayType	Type of array, can be "450k" or "EPIC"

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:

`methylation M value ~ contPheno_char + 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 regions

Examples

```
data(betasChr22_df)
data(pheno_df)

CpGsInfoOneRegion(
  regionName_char = "chr22:19709548-19709755",
  betas_df = betasChr22_df,
  pheno_df, contPheno_char = "stage",
  covariates_char = c("age.brain", "sex"),
```

```
  arrayType = "450k"
)

# not adjusting for covariates
CpGsInfoOneRegion(
  regionName_char = "chr22:18267969-18268249",
  betas_df = betasChr22_df,
  pheno_df, contPheno_char = "stage",
  covariates_char = NULL
)
```

`CreateCpGsRegions` *Create a list object with class CpGsRegions*

Description

Create a list object with class CpGsRegions

Usage

```
CreateCpGsRegions(CpGs_ls)
```

Arguments

`CpGs_ls`	a list where each item is a character vector of CpGs IDs in a region. Each vector should be named with the region name.

Value

A list object with class CpGsRegions

Examples

```
CpGsChr22_ls <- readRDS(
                   system.file ("extdata",
                                "CpGislandsChr22_ex.RDS",
                                package = 'coMethDMR',
                                mustWork = TRUE
                               )
)
CreateCpGsRegions(CpGsChr22_ls)
```

CreateParallelWorkers *Create a Parallel Computing Cluster*

Description

This function is a wrapper for the SnowParam and MulticoreParam constructor functions.

Usage

```
CreateParallelWorkers(nCores, ...)
```

Arguments

nCores	The number of computing cores to amke 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

```
workers_cl <- CreateWorkers(nCores = 4)
```

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

Description

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

Usage

```
CreateRdrop(data, method = c("pearson", "spearman"))
```

Arguments

data	a dataframe with rownames = sample IDs, column names = CpG IDs.
method	method for computing correlation, can be "pearson" or "spearman"

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.

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

```
data(betaMatrix_ex1)

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

FindComethylatedRegions

Find contiguous comethylated regions based on output file from function MarkComethylatedCpGs

---

Description

Find contiguous comethylated regions based on output file from function MarkComethylatedCpGs

Usage

```
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, indicates minimum nubmer of CpGs for output genomic regions

Value

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

Examples

```
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, arrayType = c("450k", "EPIC"))
```

Arguments

regionName_char	character string with location information for one region in this format: "chrxx:xxxxxx-xxxxxx"
arrayType	Type of array, 450k or EPIC

Value

vector of CpG probe IDs mapped to the genomic region

Examples

```
GetCpGsInRegion(
  regionName_char = "chr22:18267969-18268249",
  arrayType = "450k"
)
```

GetResiduals *Get Residuals*

Description

Get Residuals

Usage

```
GetResiduals(dnam, betaToM = TRUE, pheno_df, covariates_char,
  nCores_int = 1L, ...)
```

Arguments

dnam	data frame or matrix of methylation values, with row names = CpG IDs, column names = sample IDs. This is often the genome-wide array data. Note that if beta values are the input here, then betaToM should be set to TRUE. If mvalues are the input, then betaToM should be set to FALSE
betaToM	indicates if converting methylation beta values to mvalues
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.

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")
)
```

---

lmmTest *Fit mixed model to methylation values in one genomic region*

Description

Fit mixed model to methylation values in one genomic region

Usage

```
lmmTest(betaOne_df, pheno_df, contPheno_char, covariates_char,
  modelType = c("randCoef", "simple"), arrayType = c("450k", "EPIC"),
  outLogFile = NULL)
```

Arguments

betaOne_df
: matrix of beta values for one genomic region, with row names = CpG IDs, 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.

arrayType
: Type of array, can be "450k" or "EPIC"

outLogFile
: Name of log file for messages of mixed model analysis

Details

This function implements a mixed model to test association between methylation values in a genomic region with a continuous phenotype.

When `randCoef` is selected, the model is

`methylation M value ~ contPheno_char + covariates_char + (1|Sample) + (contPheno_char|CpG)`. The last term specifies random intercept and slope for each CpG.

When `simple` is selected, the model is

`methylation M value ~ contPheno_char + covariates_char + (1|Sample)`

In our simulation studies, we found both models are conservative, so p-values are estimated from normal distributions instead of t-distributions.

Value

A dataframe with one row for association result of one region: `Estimate`, `StdErr`, and `pvalue` for the association of methylation values in the genomic region tested vs. continuous phenotype `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"
)
```

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

Description

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

Usage

```
lmmTestAllRegions(betas, region_ls, pheno_df, contPheno_char,
  covariates_char, modelType = c("randCoef", "simple"),
  arrayType = c("450k", "EPIC"), 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, 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.
arrayType	Type of array, can be "450k" or "EPIC"
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).
...	Dots for additional arguments passed to the cluster constructor. See CreateParallelWorkers for more information.

Details

This function implements a mixed model to test association between methylation values in a genomic region with a continuous phenotype.

When randCoef is selected, the model is

methylation M value ~ contPheno_char + covariates_char + (1|Sample) + (contPheno_char|CpG). The last term specifies both random intercept and slope for each CpG.

When simple is selected, the model is

methylation M value ~ contPheno_char + covariates_char + (1|Sample)

In our simulation studies, we found both models are conservative, so p-values are estimated from normal distributions instead of t-distributions.

For the results of mixed models, note that

(1) When mixed model failed to converge, p-value for mixed model is set to 1.

(2) When 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 variabilities in data to estimate the random effects. In this case, mixed model reduces to a fixed effects model. The p-values for these regions are still valid.

Value

(1) output file: a .csv file with 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).

(2) log file: a .txt file that includes messages for mixed model fitting

Examples

```
data(betasChr22_df)

data(pheno_df)

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

coMeth_ls <- CoMethAllRegions(
  dnam = betasChr22_df,
  betaToM = TRUE,
  CpGs_ls = CpGisland_ls,
  arrayType = "450k",
  rDropThresh_num = 0.4,
  returnAllCpGs = FALSE
)


results <- lmmTestAllRegions(
  betas = betasChr22_df,
  region_ls = coMeth_ls,
  pheno_df,
  contPheno_char = "stage",
  covariates_char = "age.brain",
  modelType = "randCoef",
  arrayType = "450k",
  # generates a log file in the current directory
  outLogFile = paste0("lmmLog_", Sys.Date(), ".txt")
)
```

MarkComethylatedCpGs *Mark CpGs in contiguous and co-methylated region*

Description

Mark CpGs in contiguous and co-methylated region

Usage

```
MarkComethylatedCpGs(betaCluster_mat, betaToM = TRUE,
  rDropThresh_num = 0.4, method = c("pearson", "spearman"))
```

Arguments

betaCluster_mat	matrix of beta values, with rownames = sample ids, 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 converting to mvalues before computing correlations
rDropThresh_num	thershold for min correlation between a cpg with sum of the rest of the CpGs
method	correlation method, can be pearson or spearman

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.

Value

A data frame with the following columns:

- CpG : CpG ID
- keep : The CpGs with keep = 1 belong to the contiguous and co-methylated region
- ind : Index for the CpGs
- 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")

 data(betaMatrix_ex2)
 MarkComethylatedCpGs(betaCluster_mat = betaMatrix_ex2, method = "pearson")

 data(betaMatrix_ex3)
 MarkComethylatedCpGs(betaCluster_mat = betaMatrix_ex3, method = "pearson")

 data(betaMatrix_ex4)
MarkComethylatedCpGs(betaCluster_mat = betaMatrix_ex4, rDropThresh_num = 0.6, method = "pearson")
```

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

genome location of the CpGs, in the format of "chrxx:xxxxxx-xxxxxx"

Examples

```
 CpGs_char <- c("cg04677227", "cg07146435", "cg11632906", "cg20214853")
CpGsOrdered_df <- OrderCpGsByLocation(CpGs_char, arrayType=c("EPIC"), output = "dataframe")
 NameRegion(CpGsOrdered_df)
```

OrderCpGsByLocation *Order CpGs by genomic location*

Description

Order CpGs by genomic location

Usage

```
OrderCpGsByLocation(CpGs_char, arrayType = c("450k", "EPIC"),
  output = c("vector", "dataframe"))
```

Arguments

CpGs_char	vector of CpGs
arrayType	Type of array, 450k or EPIC
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, arrayType=c("EPIC"), 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

```
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 this format: "chrxx:xxxxxx-xxxxxx"

Value

genomic regions in GRanges format

Examples

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

`WriteCloseByAllRegions` *Extract clusters of CpG probes located closely*

Description

Extract clusters of CpG probes located closely

Usage

```
WriteCloseByAllRegions(fileName, regions, arrayType = c("450k", "EPIC"),
  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
`arrayType`	Type of array, can be "450k" or "EPIC"
`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 `maxGap` = 200 and `minCpGs` = 3, we already calculated the clusters of CpGs. They are saved in folder `/inst/extdata/`.

Value

a file with the genomic regions containing CpGs located closely within each inputing pre-defined genomic region

Examples

```
data(regions)

WriteCloseByAllRegions(
   regions = regions, arrayType = "EPIC", maxGap = 50,
   minCpGs = 3, fileName = "closeByRegions.rds"
)
```

Index