Package 'EpiMix'

April 3, 2023

Title EpiMix: an integrative tool for the population-level analysis of DNA methylation

Version 1.0.1

Description
EpiMix is a comprehensive tool for the integrative analysis of high-throughput DNA methylation data and gene expression data. EpiMix enables automated data downloading (from TCGA or GEO), preprocessing, methylation modeling, interactive visualization and functional annotation.To identify hypo- or hypermethylated CpG sites across physiological or pathological conditions, EpiMix uses a beta mixture modeling to identify the methylation states of each CpG probe and compares the methylation of the experimental group to the control group.The output from EpiMix is the functional DNA methylation that is predictive of gene expression. EpiMix incorporates specialized algorithms to identify functional DNA methylation at various genetic elements, including proximal cis-regulatory elements of protein-coding genes, distal enhancers, and genes encoding microRNAs and lncRNAs.

Depends R (>= 4.2.0), EpiMix.data (>= 0.99.2)

License GPL-3

Encoding UTF-8

Imports AnnotationHub, AnnotationDbi, Biobase, biomaRt, data.table, doParallel, doSNOW, downloader, dplyr, ELMER.data, ExperimentHub, foreach, GenomeInfoDb, GenomicFeatures, GenomicRanges, GEOquery, ggplot2, graphics, grDevices, impute, IRanges, limma, methods, parallel, plyr, progress, R.matlab, RColorBrewer, RCurl, rlang, RPMM, S4Vectors, stats, SummarizedExperiment, tibble, tidyr, utils

Suggests BiocStyle, clusterProfiler, karyoploteR, knitr, org.Hs.eg.db, regioneR, Seurat, survival, survminer, TxDb.Hsapiens.UCSC.hg19.knownGene, RUnit, BiocGenerics

biocViews Software, Epigenetics, Preprocessing, DNAMethylation, GeneExpression, DifferentialMethylation

RoxygenNote 7.2.3

VignetteBuilder knitr

BugReports https://github.com/gevaertlab/EpiMix/issues

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

git_branch RELEASE_3_16

git_last_commit 6a1aecd

git_last_commit_date 2023-02-14

Date/Publication 2023-04-03

Author Yuanning Zheng [aut, cre],
John Jun [aut],
Olivier Gevaert [aut]

Maintainer Yuanning Zheng <eric2021@stanford.edu>

R topics documented:

addDistNearestTSS *Calculate the distance between probe and gene TSS*

Description

Calculate the distance between probe and gene TSS

Usage

```
addDistNearestTSS(data, NearGenes, genome, met.platform, cores = 1)
```

Arguments

`data`	A multi Assay Experiment with both DNA methylation and gene Expression objects
`NearGenes`	A list or a data frame with the pairs gene probes
`genome`	Which genome build will be used: hg38 (default) or hg19.
`met.platform`	DNA methyaltion platform to retrieve data from: EPIC or 450K (default)
`cores`	Number fo cores to be used. Deafult: 1

Value

a dataframe of nearest genes with distance to TSS.

addGeneNames *The addGeneNames function*

Description

Given a dataframe with a column of probe names, add the gene names

Usage

```
addGeneNames(df_data, ProbeAnnotation)
```

Arguments

`df_data` a dataframe with a column named Probe

`ProbeAnnotation` a dataframe with ProbeAnnotation, including one column named 'probe' and another column named 'gene'

Value

a dataframe with added gene names

`calcDistNearestTSS` *Calculate distance from region to nearest TSS*

Description

Idea For a given region R linked to X genes G merge R with nearest TSS for G (multiple) this will increse nb of lines i.e R1 - G1 - TSS1 - DIST1 R1 - G1 - TSS2 - DIST2 To vectorize the code: make a granges from left and onde from right and find distance collapse the results keeping min distance for equals values

Usage

```
calcDistNearestTSS(links, TRange, tssAnnot)
```

Arguments

`links` Links to calculate the distance

`TRange` Genomic coordinates for Tartget region

`tssAnnot` TSS annotation

Value

dataframe of genomic distance from TSS

Author(s)

Tiago C. Silva

`ClusterProbes` *The ClusterProbes function*

Description

This function uses the annotation for Illumina methylation arrays to map each probe to a gene. Then, for each gene, it clusters all its CpG sites using hierchical clustering and Pearson correlation as distance and complete linkage. If data for normal samples is provided, only overlapping probes between cancer and normal samples are used. Probes with SNPs are removed. This function is prepared to run in parallel if the user registers a parallel structure, otherwise it runs sequentially. This function also cleans up the sample names, converting them to the 12 digit format.

Usage

```
ClusterProbes(MET_data, ProbeAnnotation, CorThreshold = 0.4)
```

Arguments

`MET_data`	data matrix for methylation.
`ProbeAnnotation`	GRange object for probe annoation.
`CorThreshold`	correlation threshold for cutting the clusters.

Value

List with the clustered data sets and the mapping between probes and genes.

`EpiMix` *The EpiMix function*

Description

EpiMix uses a model-based approach to identify functional changes DNA methylation that affect gene expression.

Usage

```
EpiMix(
  methylation.data,
  gene.expression.data,
  sample.info,
  group.1,
  group.2,
  mode = "Regular",
  promoters = FALSE,
  correlation = "negative",
  met.platform = "HM450",
  genome = "hg38",
  cluster = FALSE,
  listOfGenes = NULL,
  filter = TRUE,
  raw.pvalue.threshold = 0.05,
  adjusted.pvalue.threshold = 0.05,
  numFlankingGenes = 20,
  roadmap.epigenome.groups = NULL,
  roadmap.epigenome.ids = NULL,
  chromatin.states = c("EnhA1", "EnhA2", "EnhG1", "EnhG2"),
  NoNormalMode = FALSE,
  cores = 1,
  MixtureModelResults = NULL,
  OutputRoot = "."
)
```

Arguments

`methylation.data`
Matrix of the DNA methylation data with CpGs in rows and samples in columns.

`gene.expression.data`
Matrix of the gene expression data with genes in rows and samples in columns.

`sample.info` Dataframe that maps each sample to a study group. Should contain two columns: the first column (named 'primary') indicates the sample names, and the second column (named 'sample.type') indicating which study group each sample belongs to (e.g.,"Cancer" vs. "Normal", "Experiment" vs. "Control"). Sample names in the 'primary' column must coincide with the column names of the methylation.data.

`group.1` Character vector indicating the name(s) for the experiment group.

`group.2` Character vector indicating the names(s) for the control group.

`mode` Character string indicating the analytic mode to model DNA methylation. Should be one of the followings: 'Regular', 'Enhancer', 'miRNA' or 'lncRNA'. Default: 'Regular'. See details for more information.

`promoters` Logic indicating whether to focus the analysis on CpGs associated with promoters (2000 bp upstream and 1000 bp downstream of the transcription start site). This parameter is only used for the Regular mode.

`correlation` Character vector indicating the expected correlation between DNA methylation and gene expression. Can be either 'negative' or 'positive'. Default: 'negative'.

`met.platform` Character string indicating the microarray type for collecting the DNA methylation data. The value should be either 'HM27', 'HM450' or 'EPIC'. Default: 'HM450'

`genome` Character string indicating the genome build version to be used for CpG annotation. Should be either 'hg19' or 'hg38'. Default: 'hg38'.

`cluster` Logic indicating whether to cluster CpG site based on methylation levels using hierarchical clustering `listOfGenes` Character vector used for filtering the genes to be evaluated.

`filter` Logic indicating whether to use a linear regression filter to pre-filter the CpGs whose methyhlation correlates with gene expression. Used in the Regular mode. Default: TRUE.

`raw.pvalue.threshold`
Numeric value indicating the threshold of the raw P value for selecting the functional CpG-gene pairs. Default: 0.05.

`adjusted.pvalue.threshold`
Numeric value indicating the threshold of the adjusted P value for selecting the function CpG-gene pairs. Default: 0.05.

`numFlankingGenes`
Numeric value indicating the number of flanking genes whose expression is to be evaluated for selecting the functional enhancers. Default: 20.

`roadmap.epigenome.groups`
(parameter used for the 'Enhancer' mode) Character vector indicating the tissue group(s) to be used for selecting the enhancers. See details for more information. Default: NULL.

`roadmap.epigenome.ids`
(parameter used for the 'Enhancer' mode) Character vector indicating the epigenome ID(s) to be used for selecting the enhancers. See details for more information. Default: NULL.

`chromatin.states`
(parameter used for the 'Enhancer' mode) Character vector indicating the chromatin states to be used for selecting the enhancers. To get the available chromatin states, please run the list.chromatin.states() function. Default: c('EnhA1', 'EnhA2', 'EnhG1', 'EnhG2').

`NoNormalMode` Logical indicating if the methylation states found in the experiment group should be compared to the control group. Default: FALSE.

`cores` Number of CPU cores to be used for computation. Default: 1.

`MixtureModelResults`
Pre-computed EpiMix results, used for generating functional probe-gene pair matrix. Default: NULL

`OutputRoot` File path to store the EpiMix result object. Default: '.' (current directory)

Details

mode: EpiMix incorporates four alternative analytic modes for modeling DNA methylation: “Regular,” “Enhancer”, “miRNA” and “lncRNA”. The four analytic modes target DNA methylation analysis on different genetic elements. The Regular mode aims to model DNA methylation at proximal cis-regulatory elements of protein-coding genes. The Enhancer mode targets DNA methylation analysis on distal enhancers. The miRNA or lncRNA mode focuses on methylation analysis of miRNA- or lncRNA-coding genes.

roadmap.epigenome.groups & roadmap.epigenome.ids:

Since enhancers are cell-type or tissue-type specific, EpiMix needs to know the reference tissues or cell types in order to select the proper enhancers. EpiMix identifies enhancers from the RoadmapEpigenomic project (Nature, PMID: 25693563), which enhancers were identified by ChromHMM in over 100 tissue and cell types. Available epigenome groups (a group of relevant cell types) or epigenome ids (individual cell types) can be obtained from the original publication (Nature, PMID: 25693563, figure 2). They can also be retrieved from the list.epigenomes() function. If both roadmap.epigenome.groups and roadmap.epigenome.ids are specified, EpiMix will select all the epigenomes from the combination of the inputs.

Value

The results from EpiMix is a list with the following components:

`MethylationDrivers`
CpG probes identified as differentially methylated by EpiMix.

`NrComponents` The number of methylation states found for each driver probe.

`MixtureStates` A list with the DM-values for each driver probe. Differential Methylation values (DM-values) are defined as the difference between the methylation mean of samples in one mixture component from the experiment group and the methylation mean in samples from the control group, for a given probe.

`MethylationStates`
Matrix with DM-values for all driver probes (rows) and all samples (columns).

`Classifications`
Matrix with integers indicating to which mixture component each sample in the experiment group was assigned to, for each probe.

`Models` Beta mixture model parameters for each driver probe.

`group.1` sample names in group.1 (experimental group).

`group.2` sample names in group.2 (control group).

`FunctionalPairs`
Dataframe with the prevalence of differential methyaltion for each CpG probe in the sample population, and fold change of mRNA expression and P values for each signifcant probe-gene pair.

Examples

```
data(MET.data)
data(mRNA.data)
data(microRNA.data)
data(lncRNA.data)
data(LUAD.sample.annotation)

# Example #1: Regular mode
EpiMixResults <- EpiMix(methylation.data = MET.data,
                        gene.expression.data = mRNA.data,
                        sample.info = LUAD.sample.annotation,
                        group.1 = 'Cancer',
                        group.2 = 'Normal',
                        met.platform = 'HM450',
                        OutputRoot = tempdir())

# Example #2: Enhancer mode
EpiMixResults <- EpiMix(methylation.data = MET.data,
                       gene.expression.data = mRNA.data,
                       sample.info = LUAD.sample.annotation,
                       mode = 'Enhancer',
                       group.1 = 'Cancer',
                       group.2 = 'Normal',
                       met.platform = 'HM450',
                       roadmap.epigenome.ids = 'E096',
                       OutputRoot = tempdir())

# Example #3: miRNA mode
EpiMixResults <- EpiMix(methylation.data = MET.data,
                       gene.expression.data = microRNA.data,
                       sample.info = LUAD.sample.annotation,
                       mode = 'miRNA',
                       group.1 = 'Cancer',
                       group.2 = 'Normal',
                       met.platform = 'HM450',
                       OutputRoot = tempdir())

# Example #4: lncRNA mode
EpiMixResults <- EpiMix(methylation.data = MET.data,
                       gene.expression.data = lncRNA.data,
                       sample.info = LUAD.sample.annotation,
                       mode = 'lncRNA',
                       group.1 = 'Cancer',
                       group.2 = 'Normal',
                       met.platform = 'HM450',
                       OutputRoot = tempdir())
```

EpiMix_PlotGene *The EpiMix_PlotGene function*

Description

plot the genomic coordinate, DM values and chromatin state for each CpG probe of a specific gene.

Usage

```
EpiMix_PlotGene(
  gene.name,
  EpiMixResults,
  met.platform = "HM450",
  roadmap.epigenome.id = "E002",
  left.gene.margin = 10000,
  right.gene.margin = 10000,
  gene.name.font = 0.7,
  show.probe.name = TRUE,
  probe.name.font = 0.6,
  plot.transcripts = TRUE,
  plot.transcripts.structure = TRUE,
  y.label.font = 0.8,
  y.label.margin = 0.1,
  axis.number.font = 0.5,
  chromatin.label.font = 0.7,
  chromatin.label.margin = 0.02
)
```

Arguments

`gene.name` character string indicating the name of the gene to be plotted.

`EpiMixResults` the resulting list object returned from the function of EpiMix.

`met.platform` character string indicating the type of the microarray where the DNA methylation data were collected. The value should be either 'HM27', 'HM450' or 'EPIC'. Default: 'HM450'

`roadmap.epigenome.id` character string indicating the epigenome id (EID) for a reference tissue or cell type. Default: 'E002'

`left.gene.margin` numeric value indicating the number of extra nucleotide bases to be plotted on the left side of the target gene. Default: 10000.

`right.gene.margin` numeric value indicating the number of extra nucleotide bases to be plotted on the right side of the target gene. Default: 10000.

`gene.name.font` numeric value indicating the font size for the gene name. Default: 0.7.

`show.probe.name` logic indicating whether to show the name(s) for each differentially methylated CpG probe. Default: TRUE

`probe.name.font` numeric value indicating the font size of the name(s) for the differentially methylated probe(s) in pixels. Default: 0.6.

`plot.transcripts` logic indicating whether to plot each individual transcript of the gene. Default: TRUE. If False, the gene will be plotted with a single rectangle, without showing the structure of individual transcripts.

`plot.transcripts.structure` logic indicating whether to plot the transcript structure (introns and exons). Non-coding exons are shown in green and the coding exons are shown in red. Default: TRUE.

`y.label.font` font size of the y axis label

`y.label.margin` distance between y axis label and y axis

`axis.number.font` font size of axis ticks and numbers

`chromatin.label.font` font size of the labels of the histone proteins

`chromatin.label.margin`
distance between the histone protein labels and axis

Details

this function requires R package dependencies: karyoploteR, TxDb.Hsapiens.UCSC.hg19.knownGene, org.Hs.eg.db

roadmap.epigenome.id: since the chromatin state is tissue or cell-type specific, EpiMix needs to know the reference tissue or cell type in order to retrieve the proper DNase-seq and histone ChIP-seq data. Available epigenome ids can be obtained from the Roadmap Epigenomic study (Nature, PMID: 25693563, figure 2). They can also be retrieved from the list.epigenomes() function.

Value

plot of the genomic coordinate, DM values and chromatin state for each CpG probe of a specific gene.

Examples

```
library(karyoploteR)
library(TxDb.Hsapiens.UCSC.hg19.knownGene)
library(org.Hs.eg.db)
library(regioneR)

data(Sample_EpiMixResults_Regular)

gene.name = 'CCND2'

roadmap.epigenome.id = 'E096'

EpiMix_PlotGene(gene.name = gene.name,
                EpiMixResults = Sample_EpiMixResults_Regular,
                met.platform = 'HM450',
                roadmap.epigenome.id = roadmap.epigenome.id)
```

`EpiMix_PlotModel` *The EpiMix_PlotModel function.*

Description

Produce the mixture model and the gene expression plots representing the EpiMix results.

Usage

```
EpiMix_PlotModel(
  EpiMixResults,
  Probe,
  methylation.data,
  gene.expression.data = NULL,
  GeneName = NULL,
  axis.title.font = 20,
  axis.text.font = 16,
  legend.title.font = 18,
  legend.text.font = 18,
  plot.title.font = 20
)
```

Arguments

`EpiMixResults` resulting list object from the EpiMix function.

`Probe` character string indicating the name of the CpG probe for which to create a mixture model plot.

`methylation.data` Matrix with the methylation data with genes in rows and samples in columns.

`gene.expression.data` Gene expression data with genes in rows and samples in columns (optional). Default: NULL.

`GeneName` character string indicating the name of the gene whose expression will be ploted with the EpiMix plot (optional). Default: NULL.

`axis.title.font` font size for the axis legend.

`axis.text.font` font size for the axis label.

`legend.title.font` font size for the legend title.

`legend.text.font` font size for the legend label.

`plot.title.font` font size for the plot title.

Details

The violin plot and the scatter plot will be NULL if the gene expression data or the GeneName is not provided

Value

A list of EpiMix plots:

`MixtureModelPlot` a histogram of the distribution of DNA methylation data

ViolinPlot	a violin plot of gene expression levels in different mixutures in the MixtureModelPlot
CorrelationPlot	a scatter plot between DNA methylation and gene expression

Examples

```
{
data(MET.data)
data(mRNA.data)
data(Sample_EpiMixResults_Regular)

probe = "cg14029001"
gene.name = "CCND3"
plots <- EpiMix_PlotModel(
                          EpiMixResults = Sample_EpiMixResults_Regular,
                          Probe = probe,
                          methylation.data = MET.data,
                          gene.expression.data = mRNA.data,
                          GeneName = gene.name
                           )
plots$MixtureModelPlot
plots$ViolinPlot
plots$CorreilationPlot
}
```

EpiMix_PlotProbe *The EpiMix_PlotProbe function*

Description

plot the genomic coordinate and the chromatin state of a specific CpG probe and the nearby genes.

Usage

```
EpiMix_PlotProbe(
  probe.name,
  EpiMixResults,
  met.platform = "HM450",
  roadmap.epigenome.id = "E002",
  numFlankingGenes = 20,
  left.gene.margin = 10000,
  right.gene.margin = 10000,
  gene.name.pos = 2,
  gene.name.size = 0.5,
  gene.arrow.length = 0.05,
  gene.line.width = 2,
  plot.chromatin.state = TRUE,
  y.label.font = 0.8,
  y.label.margin = 0.1,
  axis.number.font = 0.5,
  chromatin.label.font = 0.7,
  chromatin.label.margin = 0.02
)
```

Arguments

`probe.name` character string indicating the CpG probe name.

`EpiMixResults` resulting list object returned from EpiMix.

`met.platform` character string indicating the type of micro-array where the DNA methylation data were collected.Can be either ’HM27’, ’HM450’ or ’EPIC’. Default: ’HM450’

`roadmap.epigenome.id` character string indicating the epigenome id (EID) for a reference tissue or cell type. Default: ’E002’

`numFlankingGenes` numeric value indicating the number of flanking genes to be plotted with the CpG probe. Default: 20 (10 gene upstream and 10 gene downstream).

`left.gene.margin` numeric value indicating the number of extra nucleotide bases to be plotted on the left side of the image. Default: 10000.

`right.gene.margin` numeric value indicating the number of extra nucleotide bases to be plotted on the right side of the image. Default: 10000.

`gene.name.pos` integer indicating the position for plotting the gene name relative to the gene structure. Should be 1 or 2 or 3 or 4, indicating bottom, left, top, and right, respectively.

`gene.name.size` numeric value indicating the font size of the gene names in pixels.

`gene.arrow.length` numeric value indicating the size of the arrow which indicates the positioning of the gene.

`gene.line.width` numeric value indicating the line width for the genes.

`plot.chromatin.state` logical indicating whether to plot the DNase-seq and histone ChIP-seq signals. Warnings: If the ’numFlankingGenes’ is a larger than 15, plotting the chromatin state may flood the internal memory.

`y.label.font` font size of the y axis label.

`y.label.margin` distance between y axis label and y axis.

`axis.number.font` font size of axis ticks and numbers.

`chromatin.label.font` font size of the labels of the histone proteins.

`chromatin.label.margin` distance between the histone protein labels and axis.

Details

this function requires additional dependencies: karyoploteR, TxDb.Hsapiens.UCSC.hg19.knownGene, org.Hs.eg.db

roadmap.epigenome.id: since the chromatin state is tissue or cell-type specific, EpiMix needs to know the reference tissue or cell type in order to retrieve the proper DNase-seq and histone ChIP-seq data. Available epigenome ids can be obtained from the Roadmap Epigenomic study (Nature, PMID: 25693563, figure 2). They can also be retrieved from the list.epigenomes() function.

Value

plot with CpG probe and nearby genes. Genes whose expression is significantly negatively associated with the methylation of the probe are shown in red, while the others are shown in black.

Examples

```
library(karyoploteR)
library(TxDb.Hsapiens.UCSC.hg19.knownGene)
library(org.Hs.eg.db)
library(regioneR)

data(Sample_EpiMixResults_Regular)

# The CpG site to plot
probe.name = 'cg00374492'

# The number of adjacent genes to be plotted
numFlankingGenes = 10

# Set up the reference cell/tissue type
roadmap.epigenome.id = 'E096'

# Generate the plot
EpiMix_PlotProbe(probe.name = probe.name,
                 EpiMixResults = Sample_EpiMixResults_Regular,
                 met.platform = 'HM450',
                 roadmap.epigenome.id = roadmap.epigenome.id,
                 numFlankingGenes = numFlankingGenes)
```

EpiMix_PlotSurvival *EpiMix_PlotSurvival function*

Description

function to plot Kaplan-meier survival curves for patients with different methylation state of a specific probe.

Usage

```
EpiMix_PlotSurvival(
  EpiMixResults,
  plot.probe,
  TCGA_CancerSite = NULL,
  clinical.df = NULL,
  font.legend = 16,
  font.x = 16,
  font.y = 16,
  font.tickslab = 14,
  legend = c(0.8, 0.9),
  show.p.value = TRUE
)
```

Arguments

`EpiMixResults` List of objects returned from the EpiMix function

`plot.probe` Character string with the name of the probe

`TCGA_CancerSite` TCGA cancer code (e.g. ’LUAD’)

`clinical.df` (If the TCGA_CancerSite parameter has been specified, this parameter is optional) Dataframe with survival information. Must contain at least three columns: ’sample.id’, ’days_to_death’, ’days_to_last_follow_up’.

`font.legend` numeric value indicating the font size of the figure legend. Default: 16

`font.x` numeric value indicating the font size of the x axis label. Default: 16

`font.y` numeric value indicating the font size of the y axis label. Default: 16

`font.tickslab` numeric value indicating the font size of the axis tick label. Default: 14

`legend` numeric vector indicating the x,y coordinate for positioning the figure legend. c(0,0) indicates bottom left, while c(1,1) indicates top right. Default: c(0.8,0.9). If ’none’, legend will be removed.

`show.p.value` logic indicating whether to show p value in the plot. P value was calculated by log-rank test. Default: TRUE.

Value

Kaplan-meier survival curve showing the survival time for patients with different methylation states of the probe.

Examples

```
library(survival)
library(survminer)

data(Sample_EpiMixResults_miRNA)

EpiMix_PlotSurvival(EpiMixResults = Sample_EpiMixResults_miRNA,
                    plot.probe = 'cg00909706',
                    TCGA_CancerSite = 'LUAD')
```

filterProbes *The filterProbes function*

Description

filter CpG sites based on user-specified conditions

Usage

```
filterProbes(
  mode,
  gene.expression.data,
  listOfGenes,
  promoters,
  met.platform,
  genome
)
```

Arguments

`mode`	analytic mode
`gene.expression.data`	matrix of gene expression data
`listOfGenes`	list of genes of interest
`promoters`	logic indicating whether to filter CpGs on promoters
`met.platform`	methylation platform
`genome`	genome build version

Value

filtered ProbeAnnotation

functionEnrich *The functionEnrich function*

Description

Perform functional enrichment analysis for the differentially methylated genes occurring in the significant CpG-gene pairs.

Usage

```
functionEnrich(
  EpiMixResults,
  methylation.state = "all",
  enrich.method = "GO",
  ont = "BP",
  simplify = TRUE,
  cutoff = 0.7,
  pvalueCutoff = 0.05,
  pAdjustMethod = "BH",
  qvalueCutoff = 0.2,
  save.dir = "."
)
```

Arguments

`EpiMixResults` List of the result objects returned from the EpiMix function.

`methylation.state` character string indicating whether to use all the differentially methylated genes or only use the hypo- or hyper-methylated genes for enrichment analysis. Can be either 'all', 'Hyper' or 'Hypo'.

`enrich.method` character string indicating the method to perform enrichment analysis, can be either 'GO' or 'KEGG'.

`ont` character string indicating the aspect for GO analysis. Can be one of 'BP' (i.e., biological process), 'MF' (i.e., molecular function), and 'CC' (i.e., cellular component) subontologies, or 'ALL' for all three.

`simplify` boolean value indicating whether to remove redundancy of enriched GO terms.

`cutoff` if simplify is TRUE, this is the threshold for similarity cutoff of the ajusted p value.

`pvalueCutoff` adjusted pvalue cutoff on enrichment tests to report

`pAdjustMethod` one of 'holm', 'hochberg', 'hommel', 'bonferroni', 'BH', 'BY', 'fdr', 'none'

`qvalueCutoff` qvalue cutoff on enrichment tests to report as significant. Tests must pass i) pvalueCutoff on unadjusted pvalues, ii) pvalueCutoff on adjusted pvalues and iii) qvalueCutoff on qvalues to be reported.

`save.dir` path to save the enrichment table.

Value

a clusterProfiler enrichResult instance

Examples

```
library(clusterProfiler)
library(org.Hs.eg.db)

data(Sample_EpiMixResults_Regular)

enrich.results <- function.enrich(
 EpiMixResults = Sample_EpiMixResults_Regular,
 enrich.method = 'GO',
 ont = 'BP',
 simplify = TRUE,
 save.dir = ''
)
```

---

generateFunctionalPairs

The generateFunctionalPairs function

Description

Wrapper function to get functional CpG-gene pairs

Usage

```
generateFunctionalPairs(
  MET_matrix,
  MET_Control,
  gene.expression.data,
  ProbeAnnotation,
  raw.pvalue.threshold,
  adjusted.pvalue.threshold,
  cores,
  mode = "Regular",
  correlation = "negative"
)
```

Arguments

`MET_matrix` matrix of methylation states

`MET_Control` beta values of control groups

`gene.expression.data`
matrix of gene expression data

`ProbeAnnotation`
dataframe of probe annotation

`raw.pvalue.threshold`
raw p value threshold

`adjusted.pvalue.threshold`
adjusted p value threshold

`cores` number of computational cores

`mode` character string indicating the analytic mode

`correlation` the expected relationship between DNAme and gene expression

Value

a dataframe of functional CpG-gene matrix

`GEO_Download_DNAMethylation` *The GEO_Download_DNAmethylation function*

Description

Download the methylation data and the associated sample phenotypic data from the GEO database.

Usage

```
GEO_Download_DNAMethylation(
  AccessionID,
  targetDirectory = ”.”,
  DownloadData = TRUE
)
```

Arguments

`AccessionID` character string indicating GEO accession number. Currently support the GEO series (GSE) data type.

`targetDirectory`
character string indicting the file path to save the data. Default: ’.’ (current directory).

`DownloadData` logical indicating whether the actual data should be downloaded (Default: TRUE). If False, the desired directory where the downloaded data should have been saved is returned.

Value

a list with two elements. The first element (’$MethylationData’) indicating the file path to the downloaded methylation data. The second element (’$PhenotypicData’) indicating the file path to the sample phenotypic data.

Examples

```
METdirectories <- GEO_Download_DNAMethylation(AccessionID = 'GSE114134',
                                              targetDirectory = tempdir())
```

GEO_Download_GeneExpression

The GEO_Download_GeneExpression function

Description

Download the gene expression data and the associated sample phenotypic data from the GEO database.

Usage

```
GEO_Download_GeneExpression(
  AccessionID,
  targetDirectory = ".",
  DownloadData = TRUE
)
```

Arguments

`AccessionID`	character string indicating the GEO accession number. Currently support the GEO series (GSE) data type.
`targetDirectory`	character string indicting the file path to save the data. Default: '.' (current directory)
`DownloadData`	logical indicating whether the actual data should be downloaded (Default: TRUE). If False, the desired directory where the downloaded data should have been saved is returned.

Value

a list with two elements. The first element ('$GeneExpressionData') indicating the file path to the downloaded methylation data. The second element ('$PhenotypicData') indicating the file path to the sample phenotypic data.

Examples

```
{
GEdirectories <- GEO_Download_GeneExpression(AccessionID = 'GSE114065',
                                             targetDirectory = tempdir())
}
```

GEO_GetSampleInfo *The GEO_GetSampleInfo function*

Description

auxiliary function to generate a sample information dataframe that indicates which study group each sample belongs to.

Usage

```
GEO_GetSampleInfo(METdirectories, group.column, targetDirectory = ".")
```

Arguments

`METdirectories`	list of the file paths to the downloaded DNA methylation data, which can be the output from the GEO_Download_DNAMethylation function.
`group.column`	character string indicating the column in the phenotypic data that defines the study group of each sample. The values in this column will be used to split the experiment and the control group.
`targetDirectory`	file path to save the output. Default: '.' (current directory)

Value

a dataframe with two columns: a 'primary' column indicating the actual sample names, a 'sample.type' column indicating the study group for each sample.

GEO_getSampleMap *the GEO_getSampleMap function*

Description

auxiliary function to generate a sample map for DNA methylation data and gene expression data

Usage

```
GEO_getSampleMap(METdirectories, GEdirectories, targetDirectory = ".")
```

Arguments

`METdirectories`	list of the file paths to the downloaded DNA methylation datasets, which can be the output from the GEO_Download_DNAMethylation function.
`GEdirectories`	list of the file paths to the downloaded gene expression datasets, which can be the output from the GEO_Download_GeneExpression function.
`targetDirectory`	file path to save the output. Default: '.' (current directory)

Value

dataframe with three columns: $assay (character string indicating the type of the experiment, can be either 'DNA methylation' or 'Gene expression'), $primary(character string indicating the actual sample names), $colnames (character string indicating the actual column names for each samples in DNA methylation data and gene expression data)

GEO_Preprocess_DNAMethylation

The GEO_Preprocess_DNAMethylation function

Description

Preprocess DNA methylation data from the GEO database.

Usage

```
GEO_Preprocess_DNAMethylation(
  methylation.data,
  met.platform = "EPIC",
  genome = "hg38",
  sample.info = NULL,
  group.1 = NULL,
  group.2 = NULL,
  sample.map = NULL,
  rm.chr = c("chrX", "chrY"),
  MissingValueThresholdGene = 0.2,
  MissingValueThresholdSample = 0.2,
  doBatchCorrection = FALSE,
  BatchData = NULL,
  batch.correction.method = "Seurat",
  cores = 1
)
```

Arguments

`methylation.data`
matrix of DNA methylation data with CpG in rows and sample names in columns.

`met.platform` character string indicating the type of the Illumina Infinium BeadChip for collecting the methylation data. Should be either 'HM450' or 'EPIC'. Default: 'EPIC'

`genome` character string indicating the genome build version for retrieving the probe annotation. Should be either 'hg19' or 'hg38'. Default: 'hg38'.

`sample.info` dataframe that maps each sample to a study group. Should contain two columns: the first column (named: 'primary') indicating the sample names, and the second column (named: 'sample.type') indicating which study group each sample belongs to (e.g., “Experiment” vs. “Control”, “Cancer” vs. “Normal”). Sample names in the ’primary’ column must coincide with the column names of the methylation.data. Please see details for more information. Default: NULL.

`group.1` character vector indicating the name(s) for the experiment group. The values must coincide with the values in the ’sample.type’ of the sample.info dataframe.Please see details for more information. Default: NULL.

`group.2` character vector indicating the names(s) for the control group. The values must coincide with the values in the ’sample.type’ of the sample.info dataframe. Please see details for more information. Default: NULL.

`sample.map` dataframe for mapping the GEO accession ID (column names) to the actual sample names. Can be the output from the GEO_getSampleMap function. Default: NULL.

`rm.chr` character vector indicating the probes on which chromosomes to be removed. Default: ’chrX’, ’chrY’.

`MissingValueThresholdGene`

threshold for missing values per gene. Genes with a percentage of NAs greater than this threshold are removed. Default: 0.3.

`MissingValueThresholdSample`

threshold for missing values per sample. Samples with a percentage of NAs greater than this threshold are removed. Default: 0.1.

`doBatchCorrection`

logical indicating whether to perform batch correction. If TRUE, the batch data need to be provided.

`BatchData` dataframe with batch information. Should contain two columns: the first column indicating the actual sample names, the second column indicating the batch. Users are expected to retrieve the batch information from the GEO on their own, but this can also be done using the GEO_getSampleInfo function with the ’group.column’ as the column indicating the batch for each sample. Defualt’: NULL.

`batch.correction.method`

character string indicating the method that will be used for batch correction. Should be either ’Seurat’ or ’Combat’. Default: ’Seurat’.

`cores` number of CPU cores to be used for batch effect correction. Defaut: 1.

Details

The data preprocessing pipeline includes: (1) eliminating samples and genes with too many NAs, imputing NAs. (2) (optional) mapping the column names of the DNA methylation data to the actual sample names based on the information from ’sample.map’. (3) (optional) removing CpG probes on the sex chromosomes or the user-defined chromosomes. (4) (optional) doing Batch correction. If both sample.info and group.1 and group.2 information are provided, the function will perform missing value estimation and batch correction on group.1 and group.2 separately. This will ensure that the true difference between group.1 and group.2 will not be obscured by missing value estimation and batch correction.

Value

DNA methylation data matrix with probes in rows and samples in columns.

Examples

```
{
data(MET.data)
data(LUAD.sample.annotation)

Preprocessed_Data <- GEO_Preprocess_DNAMethylation(MET.data,
                                                   met.platform = 'HM450',
                                                   sample.info = LUAD.sample.annotation,
                                                   group.1 = 'Cancer',
                                                   group.2 = 'Normal')

}
```

`GEO_Preprocess_GeneExpression`

The GEO_Preprocess_GeneExpression function

Description

Preprocess the gene expression data from the GEO database.

Usage

```
GEO_Preprocess_GeneExpression(
  gene.expression.data,
  sample.info = NULL,
  group.1 = NULL,
  group.2 = NULL,
  sample.map = NULL,
  MissingValueThresholdGene = 0.3,
  MissingValueThresholdSample = 0.1,
  doBatchCorrection = FALSE,
  BatchData = NULL,
  batch.correction.method = "Seurat",
  cores = 1
)
```

Arguments

`gene.expression.data`
a matrix of gene expression data with gene in rows and samples in columns.

`sample.info`
dataframe that maps each sample to a study group. Should contain two columns: the first column (named: ’primary’) indicating the sample names, and the second column (named: ’sample.type’) indicating which study group each sample belongs to (e.g., “Experiment” vs. “Control”, “Cancer” vs. “Normal”). Sample names in the ’primary’ column must coincide with the column names of the methylation.data. Please see details for more information. Default: NULL.

`group.1` character vector indicating the name(s) for the experiment group. The values must coincide with the values in the ’sample.type’ of the sample.info dataframe.Please see details for more information. Default: NULL.

`group.2` character vector indicating the names(s) for the control group. The values must coincide with the values in the ’sample.type’ of the sample.info dataframe. Please see details for more information. Default: NULL.

`sample.map` dataframe for mapping the GEO accession ID (column names) to the actual sample names. Can be the output from the GEO_getSampleMap function. Default: NULL.

`MissingValueThresholdGene`
threshold for missing values per gene. Genes with a percentage of NAs greater than this threshold are removed. Default is 0.3.

`MissingValueThresholdSample`
threshold for missing values per sample. Samples with a percentage of NAs greater than this threshold are removed. Default is 0.1.

`doBatchCorrection`
logical indicating whether to perform batch correction. If TRUE, the batch data need to be provided.

`BatchData` dataframe with batch information. Should contain two columns: the first column indicating the actual sample names, the second column indicating the batch. Users are expected to retrieve the batch information from GEO on their own, but this can also be done using the GEO_getSampleInfo function with the ’group.column’ as the column indicating the batch for each sample. Defualt’: NULL.

`batch.correction.method`
character string indicating the method that be used for batch correction. Should be either ’Seurat’ or ’Combat’. Default: ’Seurat’.

`cores` number of CPU cores to be used for batch effect correction. Default: 1

Details

The preprocessing pipeline includes: (1) eliminating samples and genes with too many NAs and imputing NAs. (2) if the gene names (rownames) in the gene expression data are ensembl_gene_ids or ensembl_transcript_ids, translate the gene names or the transcript names to human gene symbols (HGNC). (3) mapping the column names of the gene expression data to the actual sample names based on the information from ’sample.map’. (4) doing batch correction.

Value

gene expression data matrix with genes in rows and samples in columns.

Examples

```
{
data(mRNA.data)
data(LUAD.sample.annotation)
Preprocessed_Data <- GEO_Preprocess_GeneExpression(gene.expression.data = mRNA.data,
                                                   sample.info = LUAD.sample.annotation,
                                                   group.1 = 'Cancer',
                                               group.2 = 'Normal')
}
```

Get.Pvalue.p *Calculate empirical Pvalue*

Description

Calculate empirical Pvalue

Usage

```
Get.Pvalue.p(U.matrix, permu)
```

Arguments

`U.matrix`	A data.frame of raw pvalue from U test. Output from .Stat.nonpara
`permu`	data frame of permutation. Output from .Stat.nonpara.permu

Value

A data frame with empirical Pvalue.

getFeatureProbe *getFeatureProbe to select probes within promoter regions or distal regions.*

Description

getFeatureProbe is a function to select the probes falling into distal feature regions or promoter regions.

This function selects the probes on HM450K that either overlap distal biofeatures or TSS promoter.

Usage

```
getFeatureProbe(
  feature = NULL,
  TSS,
  genome = "hg38",
  met.platform = "HM450",
  TSS.range = list(upstream = 2000, downstream = 2000),
  promoter = FALSE,
  rm.chr = NULL
)
```

Arguments

`feature` A GRange object containing biofeature coordinate such as enhancer coordinates. If NULL only distal probes (2Kbp away from TSS will be selected) feature option is only usable when promoter option is FALSE.

`TSS` A GRange object contains the transcription start sites. When promoter is FALSE, Union.TSS in **ELMER.data** will be used for default. When promoter is TRUE, UCSC gene TSS will be used as default (see detail). User can specify their own preference TSS annotation.

`genome` Which genome build will be used: hg38 (default) or hg19.

`met.platform` DNA methyaltion platform to retrieve data from: EPIC or 450K (default)

`TSS.range` A list specify how to define promoter regions. Default is upstream =2000bp and downstream=2000bp.

`promoter` A logical.If TRUE, function will ouput the promoter probes. If FALSE, function will ouput the distal probes overlaping with features. The default is FALSE.

`rm.chr` A vector of chromosome need to be remove from probes such as chrX chrY or chrM

Details

In order to get real distal probes, we use more comprehensive annotated TSS by both GENCODE and UCSC. However, to get probes within promoter regions need more accurate annotated TSS such as UCSC. Therefore, there are different settings for promoter and distal probe selection. But user can specify their own favorable TSS annotation. Then there won't be any difference between promoter and distal probe selection. @return A GRanges object contains the coordinate of probes which locate within promoter regions or distal feature regions such as union enhancer from REMC and FANTOM5. @usage getFeatureProbe(feature, TSS, TSS.range = list(upstream = 2000, downstream = 2000), promoter = FALSE, rm.chr = NULL)

Value

A GRange object containing probes that satisfy selecting critiria.

`getMethStates_Helper` *The getMethStates_Helper function*

Description

helper function to determine the methylation state based on DM values

Usage

```
getMethStates_Helper(DMValues)
```

Arguments

`DMValues` a character vector indicating the DM values of a CpG site

Value

a character string incdicating the methylation state of the CpG

`GetNearGenes` *GetNearGenes to collect nearby genes for one locus.*

Description

GetNearGenes is a function to collect equal number of gene on each side of one locus. It can receite either multi Assay Experiment with both DNA methylation and gene Expression matrix and the names of probes to select nearby genes, or it can receive two granges objects TRange and geneAnnot.

Usage

```
GetNearGenes(
  data = NULL,
  probes = NULL,
  geneAnnot = NULL,
  TRange = NULL,
  numFlankingGenes = 20
)
```

Arguments

`data` A multi Assay Experiment with both DNA methylation and gene Expression objects

`probes` Name of probes to get nearby genes (it should be rownames of the DNA methylation object in the data argument object)

`geneAnnot` A GRange object or Summarized Experiment object that contains coordinates of promoters for human genome.

`TRange` A GRange object or Summarized Experiment object that contains coordinates of a list of targets loci.

`numFlankingGenes`
A number determines how many gene will be collected totally. Then the number devided by 2 is the number of genes collected from each side of targets (number shoule be even) Default to 20.

Value

A data frame of nearby genes and information: genes' IDs, genes' symbols, distance with target and side to which the gene locate to the target.

References

Yao, Lijing, et al. "Inferring regulatory element landscapes and transcription factor networks from cancer methylomes." Genome biology 16.1 (2015): 1.

getProbeAnnotation *The getProbeAnnotation function*

Description

Helper function to get the probe annotation based on mode

Usage

```
getProbeAnnotation(mode, met.platform, genome)
```

Arguments

`mode`	analytic mode
`met.platform`	methylation platform
`genome`	genome build version

Value

a ProbeAnnotation dataframe consisting of two columns: probe, gene

getRegionNearGenes *Identifies nearest genes to a region*

Description

Auxiliary function for GetNearGenes This will get the closest genes (n=numFlankingGenes) for a target region (TRange) based on a genome of refenrece gene annotation (geneAnnot). If the transcript level annotation (tssAnnot) is provided the Distance will be updated to the distance to the nearest TSS.

Usage

```
getRegionNearGenes(
  TRange = NULL,
  numFlankingGenes = 20,
  geneAnnot = NULL,
  tssAnnot = NULL
)
```

Arguments

`TRange`	A GRange object contains coordinate of targets.
`numFlankingGenes`	A number determine how many gene will be collected from each
`geneAnnot`	A GRange object contains gene coordinates of for human genome.
`tssAnnot`	A GRange object contains tss coordinates of for human genome.

Value

A data frame of nearby genes and information: genes' IDs, genes' symbols,

Author(s)

Tiago C Silva (maintainer: tiagochst@usp.br)

GetSurvivalProbe *The GetSurvivalProbe function*

Description

Get probes whose methylation state is predictive of patient survival

Usage

```
GetSurvivalProbe(
  EpiMixResults,
  TCGA_CancerSite = NULL,
  clinical.data = NULL,
  raw.pval.threshold = 0.05,
  p.adjust.method = "none",
  adjusted.pval.threshold = 0.05,
  OutputRoot = ""
)
```

Arguments

`EpiMixResults` List of objects returned from the EpiMix function

`TCGA_CancerSite` String indicating the TCGA cancer code (e.g. 'LUAD')

`clinical.data` (If the TCGA_CancerSite is specified, this parameter is optional) Dataframe with survival information. Must contain at least three columns: 'sample.id', 'days_to_death', 'days_to_last_follow_up'.

`raw.pval.threshold` numeric value indicting the raw p value threshold for selecting the survival predictive probes. Survival time is compared by log-rank test. Default: 0.05

`p.adjust.method` character string indicating the statistical method for adjusting multiple comparisons, can be either of 'holm', 'hochberg', 'hommel', 'bonferroni', 'BH', 'BY', 'fdr', 'none'. Default: 'fdr'

`adjusted.pval.threshold` numeric value indicting the adjusted p value threshold for selecting the survival predictive probes. Default: 0.05

`OutputRoot` path to save the output. If not null, the return value will be saved as 'Survival)Probes.csv'.

Value

a dataframe with probes whose methylation state is predictive of patient survival and the p value.

Examples

```
library(survival)

data('Sample_EpiMixResults_miRNA')

survival.CpGs <- GetSurvivalProbe(EpiMixResults = Sample_EpiMixResults_miRNA,
                                  TCGA_CancerSite = 'LUAD')
```

---

getTSS

getTSS to fetch GENCODE gene annotation (transcripts level) from Bioconductor package biomaRt If upstream and downstream are specified in TSS list, promoter regions of GENCODE gene will be generated.

Description

getTSS to fetch GENCODE gene annotation (transcripts level) from Bioconductor package biomaRt If upstream and downstream are specified in TSS list, promoter regions of GENCODE gene will be generated.

Usage

```
getTSS(genome = "hg38", TSS = list(upstream = NULL, downstream = NULL))
```

Arguments

`genome`	Which genome build will be used: hg38 (default) or hg19.
`TSS`	A list. Contains upstream and downstream like TSS=list(upstream, downstream). When upstream and downstream is specified, coordinates of promoter regions with gene annotation will be generated.

Value

GENCODE gene annotation if TSS is not specified. Coordinates of GENCODE gene promoter regions if TSS is specified.

Author(s)

Lijing Yao (maintainer: lijingya@usc.edu)

`MethylMix_Predict` *The MethylMix_Predict function*

Description

Given a new data set with methylation data, this function predicts the mixture component for each new sample and driver gene. Predictions are based on posterior probabilities calculated with MethylMix'x fitted mixture model.

Usage

```
MethylMix_Predict(newBetaValuesMatrix, MethylMixResult)
```

Arguments

`newBetaValuesMatrix`	Matrix with new observations for prediction, genes/cpg sites in rows, samples in columns. Although this new matrix can have a different number of genes/cpg sites than the one provided as METcancer when running MethylMix, naming of genes/cpg sites should be the same.
`MethylMixResult`	Output object from MethylMix

Value

A matrix with predictions (indices of mixture component), driver genes in rows, new samples in columns

`predictOneGene` *The predictOneGene function*

Description

Auxiliar function. Given a new vector of beta values, this function calculates a matrix with posterior prob of belonging to each mixture commponent (columns) for each new beta value (rows), and return the number of the mixture component with highest posterior probabilit

Usage

```
predictOneGene(newVector, mixtureModel)
```

Arguments

`newVector`	vector with new beta values
`mixtureModel`	beta mixture model object for the gene being evaluated.

Value

A matrix with predictions (indices of mixture component), driver genes in rows, new samples in columns

`removeDuplicatedGenes` *The removeDuplicatedGenes function*

Description

sum up the transcript expression values if a gene has multiple transcripts

Usage

```
removeDuplicatedGenes(GEN_data)
```

Arguments

`GEN_data`	gene expression data matrix

Value

gene expression data matrix with duplicated genes removed

`TCGA_Download_DNAmethylation` *The TCGA_Download_DNAmethylation function*

Description

Download DNA methylation data from TCGA.

Usage

```
TCGA_Download_DNAmethylation(CancerSite, TargetDirectory, downloadData = TRUE)
```

Arguments

`CancerSite`	character of length 1 with TCGA cancer code.
`TargetDirectory`	character with directory where a folder for downloaded files will be created.
`downloadData`	logical indicating if data should be downloaded (default: TRUE). If false, the url of the desired data is returned.

Value

list with paths to downloaded files for both 27k and 450k methylation data.

Examples

```
METdirectories <- TCGA_Download_DNAmethylation(CancerSit = 'OV', TargetDirectory = tempdir())
```

`TCGA_Download_GeneExpression`

The TCGA_Download_GeneExpression function

Description

Download gene expression data from TCGA.

Usage

```
TCGA_Download_GeneExpression(
  CancerSite,
  TargetDirectory,
  mode = "Regular",
  downloadData = TRUE
)
```

Arguments

`CancerSite`	character string indicating the TCGA cancer code.
`TargetDirectory`	character with directory where a folder for downloaded files will be created.
`mode`	character string indicating whether we should download the gene expression data for miRNAs or lncRNAs, instead of for protein-coding genes. See details for more information.
`downloadData`	logical indicating if the data should be downloaded (default: TRUE). If False, the url of the desired data is returned.

Details

mode: when mode is set to 'Regular', this function downloads the level 3 RNAseq data (file tag 'mRNAseq_Preprocess.Level_3'). Since there is not enough RNAseq data for OV and GBM, the micro array data is downloaded. If you plan to run the EpiMix on miRNA- or lncRNA-coding genes, please specify the 'mode' parameter to 'miRNA' or 'lncRNA'.

Value

list with paths to downloaded files for gene expression.

Examples

```
# Example #1 : download regular gene expression data for ovarian cancer
GEdirectories <- TCGA_Download_GeneExpression(CancerSite = 'OV', TargetDirectory = tempdir())

# Example #2 : download miRNA expression data for ovarian cancer
GEdirectories <- TCGA_Download_GeneExpression(CancerSite = 'OV',
                                              TargetDirectory = tempdir(),
                                              mode = 'miRNA')

# Example #3 : download lncRNA expression data for ovarian cancer
GEdirectories <- TCGA_Download_GeneExpression(CancerSite = 'OV',
                                               TargetDirectory = tempdir(),
                                               mode = 'lncRNA')
```

TCGA_GetData *The TCGA_GetData function*

Description

This function wraps the functions for downloading, pre-processing and analysis of the DNA methylation and gene expression data from the TCGA project.

Usage

```
TCGA_GetData(
  CancerSite,
  mode = "Regular",
  outputDirectory = ".",
  doBatchCorrection = FALSE,
  batch.correction.method = "Seurat",
  roadmap.epigenome.ids = NULL,
  roadmap.epigenome.groups = NULL,
  forceUse450K = FALSE,
  cores = 1
)
```

Arguments

`CancerSite` character string indicating the TCGA cancer code. The information can be found at: https://gdc.cancer.gov/resources-tcga-users/tcga-code-tables/tcga-study-abbreviations

`mode` character string indicating the analytic mode to model DNA methylation. Should be one of the followings: ’Regular’, ’Enhancer’, ’miRNA’ or ’lncRNA’. Default: ’Regular’. See details for more information.

`outputDirectory` character string indicating the file path to save the output.

`doBatchCorrection`
: logical indicating whether to do batch effect correction during preprocessing. Default: False.

`batch.correction.method`
: character string indicating the method to perform batch effect correction. The value should be either 'Seurat' or 'Combat'. Seurat is much fatster than the Combat. Default: 'Seurat'.

`roadmap.epigenome.ids`
: character vector indicating the epigenome ID(s) to be used for selecting enhancers. See details for more information. Default: NULL.

`roadmap.epigenome.groups`
: character vector indicating the tissue group(s) to be used for selecting enhancers. See details for more information. Default: NULL.

`forceUse450K` logic indicating whether force to use only 450K methylation data. Default: FALSE

`cores` Number of CPU cores to be used for computation.

Details

mode: EpiMix incorporates four alternative analytic modes for modeling DNA methylation: “Regular,” “Enhancer”, “miRNA” and “lncRNA”. The four analytic modes target DNA methylation analysis on different genetic elements. The Regular mode aims to model DNA methylation at proximal cis-regulatory elements of protein-coding genes. The Enhancer mode targets DNA methylation analysis on distal enhancers. The miRNA or lncRNA mode focuses on methylation analysis of miRNA- or lncRNA-coding genes.

roadmap.epigenome.groups & roadmap.epigenome.ids:

Since enhancers are cell-type or tissue-type specific, EpiMix needs to know the reference tissues or cell types in order to select proper enhancers. EpiMix identifies enhancers from the RoadmapEpigenomic project (Nature, PMID: 25693563), in which enhancers were identified by ChromHMM in over 100 tissue and cell types. Available epigenome groups (a group of relevant cell types) or epigenome ids (individual cell types) can be obtained from the original publication (Nature, PMID: 25693563, figure 2). They can also be retrieved from the list.epigenomes() function. If both roadmap.epigenome.groups and roadmap.epigenome.ids are specified, EpiMix will select all the epigenomes from the combination of the inputs.

Value

The results from EpiMix is a list with the following components:

`MethylationDrivers`
: CpG probes identified as differentially methylated by EpiMix.

`NrComponents` The number of methylation states found for each driver probe.

`MixtureStates` A list with the DM-values for each driver probe. Differential Methylation values (DM-values) are defined as the difference between the methylation mean of samples in one mixture component from the experiment group and the methylation mean in samples from the control group, for a given probe.

`MethylationStates`
: Matrix with DM-values for all driver probes (rows) and all samples (columns).

`Classifications`
: Matrix with integers indicating to which mixture component each sample in the experiment group was assigned to, for each probe.

`Models` Beta mixture model parameters for each driver probe.

`group.1` sample names in group.1 (experimental group).

`group.2` sample names in group.2 (control group).

`FunctionalPairs`
: Dataframe with the prevalence of differential methyaltion for each CpG probe in the sample population, and fold change of mRNA expression and P values for each signifcant probe-gene pair.

Examples

```
# Example #1 - Regular mode
EpiMixResults <- TCGA_GetData(CancerSite = 'LUAD',
                              outputDirectory = tempdir(),
                              cores = 8)

# Example #2 - Enhancer mode
EpiMixResults <- TCGA_GetData(CancerSite = 'LUAD',
                              mode =  'Enhancer',
                              roadmap.epigenome.ids = 'E097',
                              outputDirectory = tempdir(),
                              cores = 8)

Example #3 - miRNA mode
EpiMixResults <- TCGA_GetData(CancerSite = 'LUAD',
                              mode = 'miRNA',
                              outputDirectory = tempdir(),
                              cores = 8)

#' Example #4 - lncRNA mode
EpiMixResults <- TCGA_GetData(CancerSite = 'LUAD',
                              mode = 'lncRNA',
                              outputDirectory = tempdir(),
                              cores = 8)
```

`TCGA_GetSampleInfo` *The TCGA_GetSampleInfo function*

Description

The TCGA_GetSampleInfo function

Usage

```
TCGA_GetSampleInfo(METProcessedData, CancerSite = "LUAD", TargetDirectory = "")
```

Arguments

`METProcessedData`
Matrix of preprocessed methylation data.

`CancerSite` Character string of TCGA study abbreviation.

`TargetDirectory`
Path to save the sample.info. Default: ”.

Details

Generate the ’sample.info’ dataframe for TCGA data.

Value

A dataframe for the sample groups. Contains two columns: the first column (named: ’primary’) indicating the sample names, and the second column (named: ’sample.type’) indicating whether each sample is a Cancer or Normal tissue.

Examples

```
{
data(MET.data)
sample.info <- TCGA_GetSampleInfo(MET.data, CancerSite = 'LUAD')
}
```

`TCGA_Preprocess_DNAmethylation` *The TCGA_Preprocess_DNAmethylation function*

Description

Pre-processes DNA methylation data from TCGA.

Usage

```
TCGA_Preprocess_DNAmethylation(
  CancerSite,
  METdirectories,
  doBatchCorrection = FALSE,
  batch.correction.method = "Seurat",
  MissingValueThreshold = 0.2,
  cores = 1,
  use450K = FALSE
)
```

Arguments

CancerSite character string indicating the TCGA cancer code.

METdirectories character vector with directories with the downloaded data. It can be the object returned by the TCGA_Download_DNAmethylation function.

doBatchCorrection
logical indicating whether to perform batch correction. Default: False.

batch.correction.method
character string indicating the method to perform batch correction. The value should be either 'Seurat' or 'Combat'. Default: 'Seurat'. Note: Seurat is much faster than the Combat.

MissingValueThreshold
numeric values indicating the threshold for removing samples or genes with missing values.Default: 0.2.

cores integer indicating the number of cores to be used for performing batch correction with Combat.

use450K logic indicating whether to force use 450K, instead of 27K data.

Details

Pre-process includes eliminating samples and genes with too many NAs, imputing NAs, and doing Batch correction. If there are samples with both 27k and 450k data, the 27k data will be used only if the sample number in the 27k data is greater than the 450k data and there is more than 50 samples in the 27k data. Otherwise, the 450k data is used and the 27k data is discarded.

Value

pre-processed methylation data matrix with CpG probe in rows and samples in columns.

Pre-processed methylation data matrix with CpG probe in rows and samples in columns.

Examples

```
METdirectories <- TCGA_Download_DNAmethylation(CancerSite = 'OV', TargetDirectory = tempdir())
METProcessedData <- TCGA_Preprocess_DNAmethylation(CancerSite = 'OV',
                                                   METdirectories = METdirectories)
```

TCGA_Preprocess_GeneExpression
The TCGA_Preprocess_GeneExpression function

Description

Pre-processes gene expression data from TCGA.

Usage

```
TCGA_Preprocess_GeneExpression(
  CancerSite,
  MAdirectories,
  mode = "Regular",
  doBatchCorrection = FALSE,
  batch.correction.method = "Seurat",
  MissingValueThresholdGene = 0.3,
  MissingValueThresholdSample = 0.1,
  cores = 1
)
```

Arguments

`CancerSite` character string indicating the TCGA cancer code.

`MAdirectories` character vector with directories with the downloaded data. It can be the object returned by the GEO_Download_GeneExpression function.

`mode` character string indicating whether the genes in the gene expression data are miRNAs or lncRNAs. Should be either 'Regular', 'Enhancer', 'miRNA' or 'lncRNA'. This value should be consistent with the same parameter in the TCGA_Download_GeneExpress function. Default: 'Regular'.

`doBatchCorrection`
logical indicating whether to perform batch effect correction. Default: False.

`batch.correction.method`
character string indicating the method to perform batch correction. The value should be either 'Seurat' or 'Combat'. Default: 'Seurat'. Seurat is much fatster than the Combat.

`MissingValueThresholdGene`
threshold for missing values per gene. Genes with a percentage of NAs greater than this threshold are removed. Default is 0.3.

`MissingValueThresholdSample`
threshold for missing values per sample. Samples with a percentage of NAs greater than this threshold are removed. Default is 0.1.

`cores` integer indicating the number of cores to be used for performing batch correction with Combat

Details

Pre-process includes eliminating samples and genes with too many NAs, imputing NAs, and doing Batch correction. If the rownames of the gene expression data are ensembl ENSG names or ENST names, the function will convert them to the human gene symbol (HGNC).

Value

pre-processed gene expression data matrix.

Examples

```
# Example #1: Preprocessing gene expression for Regular mode

 GEdirectories <- TCGA_Download_GeneExpression(CancerSite = 'OV',
                                               TargetDirectory = tempdir())
 GEProcessedData <- TCGA_Preprocess_GeneExpression(CancerSite = 'OV',
                                                   MAdirectories = GEdirectories)

# Example #2: Preprocessing gene expression for miRNA mode

 GEdirectories <- TCGA_Download_GeneExpression(CancerSite = 'OV',
                                               TargetDirectory = tempdir(),
                                               mode = 'miRNA')

 GEProcessedData <- TCGA_Preprocess_GeneExpression(CancerSite = 'OV',
                                                   MAdirectories = GEdirectories,
                                                   mode = 'miRNA')

# Example #3: Preprocessing gene expression for lncRNA mode

 GEdirectories <- TCGA_Download_GeneExpression(CancerSite = 'OV',
                                               TargetDirectory = tempdir(),
                                               mode = 'lncRNA')

 GEProcessedData <- TCGA_Preprocess_GeneExpression(CancerSite = 'OV',
                                                   MAdirectories = GEdirectories,
                                                   mode = 'lncRNA')
```

TCGA_Select_Dataset *The TCGA_Select_Dataset function*

Description

internal function to select which MET dataset to use

Usage

```
TCGA_Select_Dataset(CancerSite, MET_Data_27K, MET_Data_450K, use450K)
```

Arguments

CancerSite	TCGA cancer code
MET_Data_27K	matrix of MET_Data_27K
MET_Data_450K	matrix of MET_Data_450K
use450K	logic indicating whether to force use 450K data

Value

the selected MET data set

---

translateMethylMixResults — *The translateMethylMixResults function*

Description

unfold clustered MethylMix results to single CpGs

Usage

```
translateMethylMixResults(MethylMixResults, probeMapping)
```

Arguments

`MethylMixResults`
list of MethylMix output

`probeMapping` dataframe of probe to gene-cluster mapping

Value

list of unfolded MethylMix results

---

validEpigenomes — *The validEpigenomes function*

Description

check user input for roadmap epigenome groups or ids

Usage

```
validEpigenomes(roadmap.epigenome.groups, roadmap.epigenome.ids)
```

Arguments

`roadmap.epigenome.groups`
epigenome groups

`roadmap.epigenome.ids`
epigenome ids

Value

a character vector of selected epigenome ids

Index