Package 'ELMER'

July 23, 2025

Title Inferring Regulatory Element Landscapes and Transcription Factor Networks Using Cancer Methylomes

Version 2.33.1

Maintainer Tiago Chedraoui Silva <tiagochst@gmail.com>

Description ELMER is designed to use DNA methylation and gene expression from a large number of samples to infere regulatory element landscape and transcription factor network in primary tissue.

Depends R (>= 3.4.0), ELMER.data (>= 2.9.3)

License GPL-3

LazyData true

VignetteBuilder knitr

Imports GenomicRanges, ggplot2, reshape, grid, grDevices, graphics, methods, parallel, stats, utils, IRanges, Seqinfo, S4Vectors, GenomicFeatures, TCGAbiolinks (>= 2.23.7), plyr, Matrix, dplyr, Gviz, ComplexHeatmap, circlize, MultiAssayExperiment, SummarizedExperiment, biomaRt, doParallel, downloader, ggrepel, lattice, magrittr, readr, scales, rvest, xml2, plotly, gridExtra, rmarkdown, stringr, tibble, tidyr, progress, purrr, reshape2, ggpubr, rtracklayer (>= 1.61.2), DelayedArray

Suggests BiocStyle, AnnotationHub, ExperimentHub, knitr, testthat, data.table, DT, GenomicInteractions, webshot, R.utils, covr, sesameData

biocViews DNAMethylation, GeneExpression, MotifAnnotation, Software, GeneRegulation, Transcription, Network

Encoding UTF-8

RoxygenNote 7.2.3

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

git_branch devel

git_last_commit fa33cdb

git_last_commit_date 2025-07-22

Repository Bioconductor 3.22

Date/Publication 2025-07-23

Author Tiago Chedraoui Silva [aut, cre],
Lijing Yao [aut],
Simon Coetzee [aut],
Nicole Gull [ctb],
Hui Shen [ctb],
Peter Laird [ctb],
Peggy Farnham [aut],
Dechen Li [ctb],
Benjamin Berman [aut]

Contents

`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

Examples

```
## Not run:
 data <- ELMER:::getdata("elmer.data.example")
 NearbyGenes <- GetNearGenes(
  data = data,
  probes = c("cg15924102", "cg24741609"),
  numFlankingGenes = 20
 )
 NearbyGenes <- addDistNearestTSS(data = data, NearGenes = NearbyGenes)

## End(Not run)
```

addMutCol *Adds mutation information to MAE*

Description

Adds mutation information to MAE

Usage

```
addMutCol(
  data,
  disease,
  genes,
  mutant_variant_classification = c("Frame_Shift_Del", "Frame_Shift_Ins",
    "Missense_Mutation", "Nonsense_Mutation", "Splice_Site", "In_Frame_Del",
    "In_Frame_Ins", "Translation_Start_Site", "Nonstop_Mutation")
)
```

Arguments

data	MAE object
disease	TCGA disease (LUSC, GBM, etc)
genes	list of genes to add information
mutant_variant_classification	List of mutant_variant_classification that will be consider a sample mutant or not.

Examples

```
## Not run:
 data <- ELMER:::getdata("elmer.data.example") # Get data from ELMER.data
 data <- ELMER:::addMutCol(data, "LUSC","TP53")

## End(Not run)
```

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

Author(s)

Tiago C. Silva

Examples

```
## Not run:
 data <- ELMER:::getdata("elmer.data.example")
  NearbyGenes <- GetNearGenes(
    data = data,
    probes = c("cg15924102", "cg24741609"),
    numFlankingGenes = 20
 )

  NearbyGenes <- ELMER:::calcDistNearestTSS(
       links = NearbyGenes,
       tssAnnot =  getTSS(genome = "hg38"),
       TRange = rowRanges(getMet(data))
  )

## End(Not run)
```

---

calculateEnrichement *Calculate motif Erichment*

Description

Calculates fisher exact test

Usage

```
calculateEnrichement(foreground, background)
```

Arguments

foreground	A nsparseMatrix object in each 1 means the motif is found in a region, 0 not.
background	A nsparseMatrix object in each 1 means the motif is found in a region, 0 not.

Examples

```
foreground <- Matrix::Matrix(sample(0:1,size = 100,replace = TRUE),
                             nrow = 10, ncol = 10,sparse = TRUE)
rownames(foreground) <- paste0("region",1:10)
colnames(foreground) <- paste0("motif",1:10)
background <- Matrix::Matrix(sample(0:1,size = 100,replace = TRUE),
                             nrow = 10, ncol = 10,sparse = TRUE)
rownames(background) <- paste0("region",1:10)
colnames(background) <- paste0("motif",1:10)
calculateEnrichement(foreground,background)
```

createBigWigDNAmetArray *Create a bigwig file for IGV visualization of DNA methylation data (Array)*

Description

Create a bigwig for IGV visualization of DNA methylation data (Array)

Usage

```
createBigWigDNAmetArray(
  data = NULL,
  genome = "hg38",
  met.platform = "450K",
  track.names = NULL,
  dir = "IGV_tracks"
)
```

Arguments

data	A matrix
genome	Which genome build will be used: hg38 (default) or hg19.
met.platform	DNA methyaltion platform to retrieve data from: EPIC or 450K (default)
track.names	Provide a list of track names (.bw) otherwise the deault is the will be samples.bw
dir	Which directory files will be saved

Author(s)

Tiago Chedraoui Silva (tiagochst at gmail.com)

Examples

```
## Not run:
data <- assay(getMet(ELMER:::getdata("elmer.data.example")))
createBigWigDNAmetArray(data = data, met.platform = "450K", genome = "hg38")

## End(Not run)
```

createIGVtrack *Create a junction track for IGV visualization of interection*

Description

Create a junction track for IGV visualization of interection

Usage

```
createIGVtrack(
  pairs,
  met.platform = "450K",
  genome = "hg38",
  filename = "ELMER_interactions.bed",
  color.track = "black",
  track.name = "junctions",
  gene.symbol = NULL,
  all.tss = TRUE
)
```

Arguments

pairs	A data frame output from getPairs function
met.platform	DNA methyaltion platform to retrieve data from: EPIC or 450K (default)
genome	Which genome build will be used: hg38 (default) or hg19.
filename	Filename (".bed")
color.track	A color for the track (i.e blue, red,#272E6A)
track.name	Track name
gene.symbol	Filter pairs to a single gene.
all.tss	A logical. If TRUE it will link probes to all TSS of a gene (transcript level), if FALSE it will link to the promoter region of a gene (gene level).

Author(s)

Tiago Chedraoui Silva (tiagochst at gmail.com)

Examples

```
 ## Not run:
data <- ELMER:::getdata("elmer.data.example")
nearGenes <-GetNearGenes(TRange=getMet(data)[c("cg00329272","cg10097755"),],
                         geneAnnot=getExp(data))
Hypo.pair <- get.pair(data=data,
                      nearGenes=nearGenes,
                      permu.size=5,
                      group.col = "definition",
                      group1 = "Primary solid Tumor",
                      group2 = "Solid Tissue Normal",
                      raw.pvalue = 0.2,
                      Pe = 0.2,
                      dir.out="./",
                      label= "hypo")
 createIGVtrack(Hypo.pair,met.platform = "450K", genome = "hg38")

## End(Not run)
```

createMAE *Construct a Multi Assay Experiment for ELMER analysis*

Description

This function will receive a gene expression and DNA methylation data objects and create a Multi Assay Experiment.

Usage

```
createMAE(
  exp,
  met,
  colData,
  sampleMap,
  linearize.exp = FALSE,
  filter.probes = NULL,
  met.na.cut = 0.2,
  filter.genes = NULL,
  met.platform = "450K",
  genome = NULL,
  save = TRUE,
  save.filename,
  TCGA = FALSE
)
```

Arguments

exp	A Summarized Experiment with one assay, or a matrix or path of rda file only containing the data. Rownames should be either Ensembl gene id (ensembl_gene_id) or gene symbol (external_gene_name)
met	A Summarized Experiment with one assay containing beta-values, a matrix or path of rda file only containing the data.
colData	A DataFrame or data.frame of the phenotype data for all participants. Must have column primary (sample ID).
sampleMap	A DataFrame or data.frame of the matching samples and colnames of the gene expression and DNA methylation matrix. This should be used if your matrix have different columns names. This object must have following columns: assay ("DNA methylation" and "Gene expression"), primary (sample ID) and colname (names of the columns of the matrix).
linearize.exp	Take log2(exp + 1) in order to linearize relation between methylation and expression
filter.probes	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. See get.feature.probe function.
met.na.cut	Define the percentage of NA that the line should have to remove the probes for humanmethylation platforms.
filter.genes	List of genes ensemble ids to filter from object
met.platform	DNA methylation platform "450K" or "EPIC"

genome	Which is the default genome to make gene information. Options hg19 and hg38
save	If TRUE, MAE object will be saved into a file named as the argument save.file if this was set, otherwise as mae_genome_met.platform.rda.
save.filename	Name of the rda file to save the object (must end in .rda)
TCGA	A logical. FALSE indicate data is not from TCGA (FALSE is default). TRUE indicates data is from TCGA and sample section will automatically filled in.

Value

A MultiAssayExperiment object

Examples

```
# NON TCGA example: matrices has different column names
gene.exp <- S4Vectors::DataFrame(
   sample1.exp = c("ENSG00000141510"=2.3,"ENSG00000171862"=5.4),
   sample2.exp = c("ENSG00000141510"=1.6,"ENSG00000171862"=2.3)
)

dna.met <- S4Vectors::DataFrame(
   sample1.met = c("cg14324200"=0.5,"cg23867494"=0.1),
   sample2.met =  c("cg14324200"=0.3,"cg23867494"=0.9)
)
sample.info <- S4Vectors::DataFrame(
  primary =  c("sample1","sample2"),
  sample.type = c("Normal", "Tumor")
 )

sampleMap <- S4Vectors::DataFrame(
    assay = c("Gene expression","DNA methylation","Gene expression","DNA methylation"),
    primary = c("sample1","sample1","sample2","sample2"),
    colname = c("sample1.exp","sample1.met","sample2.exp","sample2.met")
)

mae <- createMAE(
    exp = gene.exp,
    met = dna.met,
    sampleMap = sampleMap,
    met.platform ="450K",
    colData = sample.info,
    genome = "hg38"
)

# You can also use sample Mapping and Sample information tables from a tsv file
# You can use the createTSVTemplates function to create the tsv files
readr::write_tsv(as.data.frame(sampleMap), path = "sampleMap.tsv")
readr::write_tsv(as.data.frame(sample.info), path = "sample.info.tsv")

mae <- createMAE(
   exp = gene.exp,
   met = dna.met,
   sampleMap = "sampleMap.tsv",
   met.platform ="450K",
   colData = "sample.info.tsv",
   genome = "hg38"
)
```

```
# NON TCGA example: matrices has same column names
gene.exp <- S4Vectors::DataFrame(sample1 = c("ENSG00000141510"=2.3,"ENSG00000171862"=5.4),
                  sample2 = c("ENSG00000141510"=1.6,"ENSG00000171862"=2.3))
dna.met <- S4Vectors::DataFrame(sample1 = c("cg14324200"=0.5,"cg23867494"=0.1),
                       sample2=  c("cg14324200"=0.3,"cg23867494"=0.9))
sample.info <- S4Vectors::DataFrame(primary =  c("sample1","sample2"),
                                    sample.type = c("Normal", "Tumor"))
sampleMap <- S4Vectors::DataFrame(
     assay = c("Gene expression","DNA methylation","Gene expression","DNA methylation"),
    primary = c("sample1","sample1","sample2","sample2"),
    colname = c("sample1","sample1","sample2","sample2")
)
mae <- createMAE(
  exp = gene.exp,
  met = dna.met,
  sampleMap = sampleMap,
  met.platform ="450K",
  colData = sample.info,
  genome = "hg38"
)

## Not run:
   # TCGA example using TCGAbiolinks
   # Testing creating MultyAssayExperiment object
   # Load library
   library(TCGAbiolinks)
   library(SummarizedExperiment)

   samples <- c(
      "TCGA-BA-4074", "TCGA-BA-4075", "TCGA-BA-4077", "TCGA-BA-5149",
       "TCGA-UF-A7JK", "TCGA-UF-A7JS", "TCGA-UF-A7JT", "TCGA-UF-A7JV"
   )

   #1) Get gene expression matrix
   query.exp <- GDCquery(
       project = "TCGA-HNSC",
       data.category = "Transcriptome Profiling",
       data.type = "Gene Expression Quantification",
       workflow.type = "STAR - Counts",
       barcode = samples
   )

   GDCdownload(query.exp)
   exp.hg38 <- GDCprepare(query = query.exp)

   # DNA Methylation
   query.met <- GDCquery(
      project = "TCGA-HNSC",
      data.category = "DNA Methylation",
      data.type = "Methylation Beta Value",
      barcode = samples,
      platform = "Illumina Human Methylation 450"
   )

   GDCdownload(query.met)
   met <- GDCprepare(query = query.met)
```

```
distal.enhancer <- get.feature.probe(
   genome = "hg19",
   met.platform = "450k"
)


mae.hg38 <- createMAE(
  exp = exp.hg38, met = met,
  TCGA = TRUE, genome = "hg38",
  filter.probes = distal.enhancer
)
values(getExp(mae.hg38))

# Consisering it is TCGA and not SE
mae.hg19.test <- createMAE(
  exp = assay(exp.hg19), met =  assay(met),
  TCGA = TRUE, genome = "hg19",
  filter.probes = distal.enhancer
)

mae.hg38 <- createMAE(
  exp = assay(exp.hg38),
  met = assay(met),
  TCGA = TRUE,
  genome = "hg38",
  filter.probes = distal.enhancer
)
values(getExp(mae.hg38))

# Consisering it is not TCGA and SE
# DNA methylation and gene expression Objects should have same sample names in columns
not.tcga.exp <- exp.hg19
colnames(not.tcga.exp) <- substr(colnames(not.tcga.exp),1,15)
not.tcga.met <- met
colnames(not.tcga.met) <- substr(colnames(not.tcga.met),1,15)

phenotype.data <- data.frame(
   row.names = colnames(not.tcga.exp),
   primary = colnames(not.tcga.exp),
   samples = colnames(not.tcga.exp),
   group = c(rep("group1",4),rep("group2",4))
)

distal.enhancer <- get.feature.probe(genome = "hg19",met.platform = "450k")
mae.hg19 <- createMAE(
   exp = not.tcga.exp,
   met =  not.tcga.met,
   TCGA = FALSE,
   filter.probes = distal.enhancer,
   genome = "hg19",
   colData = phenotype.data
)

## End(Not run)
createMAE
```

`createMotifRelevantTfs` *Get family of transcription factors*

Description

This will output a list each TF motif and TFs that binding the motis. Multiple TFs may recognize a same motif such as TF family. The association between each motif famil and transcription factor was created using the (HOCOMOCO)[https://hocomoco11.autosome.org/human/mono?full=true] which TF structural families was created according to TFClass [@wingender2014tfclass] This data is stored as a list whose elements are motifs and contents for each element are TFs which recognize the same motif that is the name of the element. This data is used in function get.TFs in **ELMER** to identify the real regulator TF whose motif is enriched in a given set of probes and expression associate with average DNA methylation of these motif sites.

Usage

```
createMotifRelevantTfs(classification = "family")
```

Arguments

`classification` Select if we will use Family classification or sub-family

Value

A list of TFs and its family members

`createSummaryDocument` *Create summary document for TCGA.pipe function*

Description

This function will create a text file with the date of the last run, which aanalysis were performed, the values of the arguments so the user can keep track

Usage

```
createSummaryDocument(
  analysis = "all",
  argument.values = "defaults",
  genome = NULL,
  mae.path = NULL,
  mode = NULL,
  direction = NULL,
  group.col = NULL,
  group1 = NULL,
  group2 = NULL,
  results.path = NULL
)
```

Arguments

analysis	Which analysis were performed
argument.values	Other argument values changed
genome	Genome of reference hg38 and hg19
mae.path	Where mae is stored
mode	Mode "supervised" or "unsupervised" used in the analysis
direction	Hypo or hyper direction
group.col	Group col
group1	Group 1
group2	Group 2
results.path	Path where the results were saved

---

createTSVTemplates *Create examples files for Sample mapping and information used in createMAE function*

Description

This function will receive the DNA methylation and gene expression matrix and will create some examples of table for the argument colData and sampleMap used in ceeateMae function.

Usage

```
createTSVTemplates(met, exp)
```

Arguments

met	DNA methylation matrix or Summarized Experiment
exp	Gene expression matrix or Summarized Experiment

Examples

```
gene.exp <- S4Vectors::DataFrame(sample1.exp = c("ENSG00000141510"=2.3,"ENSG00000171862"=5.4),
                  sample2.exp = c("ENSG00000141510"=1.6,"ENSG00000171862"=2.3))
dna.met <- S4Vectors::DataFrame(sample1.met = c("cg14324200"=0.5,"cg23867494"=0.1),
                        sample2.met =  c("cg14324200"=0.3,"cg23867494"=0.9))
createTSVTemplates(met = dna.met, exp = gene.exp)
```

ELMER *ELMER (Enhancer Linking by Methylation/Expression Relationships)*

Description

ELMER is designed to use DNA methylation and gene expression from a large number of samples to infere regulatory element landscape and transcription factor network in primary tissue.

`findMotifRegion` *Use Hocomoco motif and homer to identify motifs in a given region*

Description

To find for each probe the know motif we will use HOMER software (http://homer.salk.edu/homer/). Homer and genome should be installed before this function is executed Step: 1 - get DNA methylation probes annotation with the regions 2 - Make a bed file from it 3 - Execute section: Finding Instance of Specific Motifs from http://homer.salk.edu/homer/ngs/peakMotifs.html to the HOCOMOCO TF motifs Also, As HOMER is using more RAM than the available we will split the files in to 100k probes. Obs: for each probe we create a winddow of 500 bp (-size 500) around it. This might lead to false positives, but will not have false negatives. The false posives will be removed latter with some statistical tests.

Usage

```
findMotifRegion(
  regions,
  output.filename = "mapped_motifs_regions.txt",
  region.size = NULL,
  genome = "hg38",
  nstep = 10000,
  cores = 1
)
```

Arguments

`regions`	A GRanges object. Names will be used as the identifier.
`output.filename`	Final file name
`region.size`	If NULL the motif will be mapped to the region. If set a window around its center will be considered. For example if region.size is 500, then +-250bp round it will be searched.
`genome`	Homer genome (hg38, hg19)
`nstep`	Number of regions to evaluate in homer, the bigger, more memory it will use at each step.
`cores`	A interger which defines the number of cores to be used in parallel process. Default is 1: no parallel process.

Examples

```
## Not run:
 # use the center of the region and +-250bp around it
 gr0 <- GRanges(Rle(c("chr2", "chr2", "chr1", "chr3"),
                    c(1, 3, 2, 4)
                    ),
                IRanges(1:10, width=10:1)
                )
 names(gr0) <- paste0("ID",c(1:10))
 findMotifRegion(regions = gr0, region.size = 500, genome = "hg38", cores = 1)
```

```
 # use the region size itself
 gr1 <- GRanges(Rle(c("chr2", "chr2", "chr1", "chr3"), c(1, 3, 2, 4)),
                IRanges(1:10, width=sample(200:1000,10)))
 names(gr1) <- paste0("ID",c(1:10))
 findMotifRegion(regions = gr0, genome = "hg38", cores = 1)

## End(Not run)
```

`get.diff.meth` *Identify hypo/hyper-methylated CpG sites between two groups (i.e. normal vs tumor samples, treated vs untreated).*

Description

get.diff.meth applys one-way t-test to identify the CpG sites that are significantly hypo/hyper-methyalated using proportional samples (defined by minSubgroupFrac option) from group 1 and group 2. The P values will be adjusted by Benjamini-Hochberg method. Option pvalue and sig.dif will be the criteria (cutoff) for selecting significant differentially methylated CpG sites. If save is TURE, two getMethdiff.XX.csv files will be generated (see detail).

Usage

```
get.diff.meth(
  data,
  diff.dir = "hypo",
  cores = 1,
  mode = "unsupervised",
  minSubgroupFrac = 0.2,
  pvalue = 0.01,
  group.col,
  min.samples = 5,
  group1,
  group2,
  test = t.test,
  sig.dif = 0.3,
  dir.out = "./",
  save = TRUE
)
```

Arguments

`data`	A multiAssayExperiment with DNA methylation and Gene Expression data. See `createMAE` function.
`diff.dir`	A character can be "hypo", "hyper" or "both", showing differential methylation direction. It can be "hypo" which is only selecting hypomethylated probes (one tailed test); "hyper" which is only selecting hypermethylated probes (one tailed test); or "both" which are probes differenly methylated (two tailed test).
`cores`	A interger which defines the number of cores to be used in parallel process. Default is 1: no parallel process.

mode	A character. Can be "unsupervised" or "supervised". If "supervised", the minSubgroupFrac argument will be set to 1 to use all samples from both groups to find the differently methylated regions. The supervised mode should be used when all samples from both groups are considered homogenous (i.e. treated vs untreated, molecular subtype A vs molecular subtype B), while unsupervised mode should be used when there is at least one group with heterogenous samples (i.e tumor samples).
minSubgroupFrac	A number ranging from 0 to 1, specifying the fraction of extreme samples from group 1 and group 2 that are used to identify the differential DNA methylation. The default is 0.2 because we typically want to be able to detect a specific (possibly unknown) molecular subtype among tumor; these subtypes often make up only a minority of samples, and 20% was chosen as a lower bound for the purposes of statistical power. If you are using pre-defined group labels, such as treated replicates vs. untreated replicated, use a value of 1.0 (Supervised mode)
pvalue	A number specifies the significant P value (adjusted P value by BH) threshold Limit for selecting significant hypo/hyper-methylated probes. Default is 0.01 If pvalue is smaller than pvalue than it is considered significant.
group.col	A column defining the groups of the sample. You can view the available columns using: colnames(MultiAssayExperiment::colData(data)).
min.samples	Minimun number of samples to use in the analysis. Default 5. If you have 10 samples in one group, minSubgroupFrac is 0.2 this will give 2 samples in the lower quintile, but then 5 will be used.
group1	A group from group.col. ELMER will run group1 vs group2. That means, if direction is hyper, get probes hypermethylated in group 1 compared to group 2.
group2	A group from group.col. ELMER will run group1 vs group2. That means, if direction is hyper, get probes hypermethylated in group 1 compared to group 2.
test	Statistical test to be used. Options: t.test (DEFAULT), wilcox.test
sig.dif	A number specifies the smallest DNA methylation difference as a cutoff for selecting significant hypo/hyper-methylated probes. Default is 0.3.
dir.out	A path specify the directory for outputs. Default is is current directory.
save	A logic. When TRUE, two getMethdiff.XX.csv files will be generated (see detail)

Details

save: When save is TRUE, function will generate two XX.csv files.The first one is named getMethdiff.hypo.probes.csv (or getMethdiff.hyper.probes.csv depends on diff.dir). The first file contains all statistic results for each probe. Based on this file, user can change different P value or sig.dir cutoff to select the significant results without redo the analysis. The second file is named getMethdiff.hypo.probes.significant.csv (or getMethdiff.hyper.probes.significant.csv depends on diff.dir). This file contains statistic results for the probes that pass the significant criteria (P value and sig.dir). When save is FALSE, a data frame R object will be generate which contains the same information with the second file.

Value

Statistics for all probes and significant hypo or hyper-methylated probes.

References

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

Examples

```
data <- ELMER:::getdata("elmer.data.example")
Hypo.probe <- get.diff.meth(data,
                            diff.dir="hypo",
                            group.col = "definition",
                            group1 = "Primary solid Tumor",
                            group2 = "Solid Tissue Normal",
                            sig.dif = 0.1) # get hypomethylated probes
Hyper.probe <- get.diff.meth(data,
                            diff.dir="hyper",
                            group.col = "definition",
                            sig.dif = 0.1) # get hypomethylated probes
```

`get.enriched.motif` *get.enriched.motif to identify the overrepresented motifs in a set of probes (HM450K) regions.*

Description

get.enriched.motif is a function make use of Probes.motif data from **ELMER.data** package to calculate the motif enrichment Odds Ratio and 95% confidence interval for a given set of probes using fisher test function, after performing the Fisher's exact test, the results for all transcription factors are corrected for multiple testing with the Benjamini-Hochberg procedure. If save is TURE, two output files will be saved: getMotif.XX.enriched.motifs.rda and getMotif.XX.motif.enrichment.csv (see detail).

Usage

```
get.enriched.motif(data, probes.motif, probes, min.motif.quality = "DS",
            background.probes, pvalue = 0.05, lower.OR = 1.1, min.incidence = 10,
                  dir.out = "./", label = NULL, save = TRUE, plot.title="")
```

Arguments

`data`	A multi Assay Experiment from `createMAE` function. If set and probes.motif/background probes are missing this will be used to get this other two arguments correctly. This argument is not require, you can set probes.motif and the backaground.probes manually.
`probes.motif`	A matrix contains motifs occurrence within probes regions. Probes.motif in **ELMER.data** will be used if probes.motif is missing (detail see Probes.motif.hg19.450K in ELMER.data).
`probes`	A vector lists the name of probes to define the set of probes in which motif enrichment OR and confidence interval will be calculated.

`min.motif.quality`
: Minimum motif quality score to consider. Possible valules: A, B, C , D, AS (A and S), BS (A, B and S), CS (A, B , C and S), DS (all - default) Description: Each PWM has a quality rating from A to D where A represents motifs with the highest confidence, and D motifs only weakly describe the pattern with a limited applications for quantitative analyses. Special S quality marks the single-box motifs (secondary motif). Source: http://hocomoco.autosome.ru/help#description_quality_score More information: http://nar.oxfordjournals.org/content/44/D1/D116.full#sec-8

`background.probes`
: A vector lists name of probes which are considered as background for motif.enrichment calculation (see detail).

`pvalue`
: FDR P-value cut off (default 0.05)

`lower.OR`
: A number specifies the smallest lower boundary of 95% confidence interval for Odds Ratio. The motif with higher lower boudnary of 95% confidence interval for Odds Ratio than the number are the significantly enriched motifs (detail see reference).

`min.incidence`
: A non-negative integer specifies the minimum incidence of motif in the given probes set. 10 is default.

`dir.out`
: A path. Specifies the directory for outputs. Default is current directory

`label`
: A character. Labels the outputs such as "hypo", "hyper"

`save`
: If save is TURE, two files will be saved: getMotif.XX.enriched.motifs.rda and getMotif.XX.motif.enrichment.csv (see detail).

`plot.title`
: Plot title. Default: no title.

Details

background.probes: For enhancer study, it is better to use probes within distal enhancer probes as background.probes. For promoter study, it is better to use probes within promoter regions as background.probes. Because enhancer and promoter have different CG content and harbors different clusters of TFs motif.

save: if save is TRUE, two files will be save on the disk. The first file is getMotif.XX.motif.enrichment.csv (XX depends on option label). This file reports the Odds Ratio and 95% confidence interval for these Odds Ratios which pass the significant cutoff (lower.OR and min.incidence). The second file is getMotif.XX.enriched.motifs.rda (XX depends on option lable). This file contains a list R object with enriched motifs as name and probes containing the enriched motif as contents. This object will be used in `get.TFs` function. if save is FALSE, the function will return a R object which is the same with second file.

Value

A list contains enriched motifs with the probes regions harboring the motif.

A list (R object) with enriched motifs as name and probes containing the enriched motif as contents. And hypo.motif.enrichment.pdf plot will be generated.

Author(s)

Lijing Yao (creator: lijingya@usc.edu)

References

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

Examples

```
probes <- c("cg00329272","cg10097755","cg08928189", "cg17153775","cg21156590",
"cg19749688","cg12590404","cg24517858","cg00329272","cg09010107",
"cg15386853", "cg10097755", "cg09247779","cg09181054","cg19371916")
  data <- tryCatch(ELMER:::getdata("elmer.data.example"), error = function(e) {
  message(e)
  data(elmer.data.example, envir = environment())
  })
bg <- rownames(getMet(data))
data(Probes.motif.hg38.450K,package = "ELMER.data")
enriched.motif <- get.enriched.motif(
   probes.motif = Probes.motif.hg38.450K,
   probes = probes,
   background.probes = bg,
   pvalue = 1,
   min.incidence = 2,
   label = "hypo"
)
# If the MAE is set, the background and the probes.motif will
# be automatically set
enriched.motif <- get.enriched.motif(
    data = data,
    min.motif.quality = "DS",
    probes=probes,
    pvalue = 1,
    min.incidence=2,
    label="hypo"
)
```

`get.feature.probe` *get.feature.probe to select probes within promoter regions or distal regions.*

Description

get.feature.probe 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

```
get.feature.probe(
  feature = NULL,
  TSS,
  genome = "hg38",
  met.platform = "450K",
  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 get.feature.probe(feature, TSS, TSS.range = list(upstream = 2000, downstream = 2000), promoter = FALSE, rm.chr = NULL)

Value

A GRange object containing probes that satisfy selecting critiria.

Examples

```
# get distal enhancer probe
## Not run:
Probe <- get.feature.probe()

## End(Not run)
# get promoter probes
## Not run:
Probe <- get.feature.probe(promoter=FALSE)

## End(Not run)
# get distal enhancer probe remove chrX chrY
Probe2 <- get.feature.probe(rm.chr=c("chrX", "chrY"))
```

get.pair *get.pair to predict enhancer-gene linkages.*

Description

get.pair is a function to predict enhancer-gene linkages using associations between DNA methylation at enhancer CpG sites and expression of 20 nearby genes of the CpG sites (see reference). Two files will be saved if save is true: getPair.XX.all.pairs.statistic.csv and getPair.XX.pairs.significant.csv (see detail).

Usage

```
get.pair(data,
         nearGenes,
         minSubgroupFrac = 0.4,
         permu.size = 10000,
         permu.dir = NULL,
         raw.pvalue = 0.001,
         Pe = 0.001,
         mode = "unsupervised",
         diff.dir = NULL,
         dir.out = "./",
         diffExp = FALSE,
         group.col,
         group1 = NULL,
         group2 = NULL,
         cores = 1,
         correlation = "negative",
         filter.probes = TRUE,
         filter.portion = 0.3,
         filter.percentage = 0.05,
         label = NULL,
         addDistNearestTSS = FALSE,
         save = TRUE)
```

Arguments

data	A multiAssayExperiment with DNA methylation and Gene Expression data. See createMAE function.
nearGenes	Can be either a list containing output of GetNearGenes function or path of rda file containing output of GetNearGenes function.
minSubgroupFrac	A number ranging from 0 to 1, specifying the fraction of extreme samples that define group U (unmethylated) and group M (methylated), which are used to link probes to genes. The default is 0.4 (the lowest quintile of samples is the U group and the highest quintile samples is the M group) because we typically want to be able to detect a specific (possibly unknown) molecular subtype among tumor; these subtypes often make up only a minority of samples, and 20% was chosen as a lower bound for the purposes of statistical power. If you are using predefined group labels, such as treated replicates vs. untreated replicated, use a value of 1.0 (Supervised mode).

permu.size — A number specify the times of permuation used in the unsupervised mode. Default is 10000.

permu.dir — A path where the output of permutation will be.

raw.pvalue — A number specify the raw p-value cutoff for defining significant pairs. Default is 0.001. It will select the significant P value cutoff before calculating the empirical p-values.

Pe — A number specify the empirical p-value cutoff for defining significant pairs. Default is 0.001

mode — A character. Can be "unsupervised" or "supervised". If unsupervised is set the U (unmethylated) and M (methylated) groups will be selected among all samples based on methylation of each probe. Otherwise U group and M group will set as the samples of group1 or group2 as described below: If diff.dir is "hypo, U will be the group 1 and M the group2. If diff.dir is "hyper" M group will be the group1 and U the group2.

diff.dir — A character can be "hypo" or "hyper", showing differential methylation direction in group 1. It can be "hypo" which means the probes are hypomethylated in group1; "hyper" which means the probes are hypermethylated in group1; This argument is used only when mode is supervised nad it should be the same value from get.diff.meth function.

dir.out — A path specify the directory for outputs. Default is current directory

diffExp — A logic. Default is FALSE. If TRUE, t test will be applied to test whether putative target gene are differentially expressed between two groups.

group.col — A column defining the groups of the sample. You can view the available columns using: colnames(MultiAssayExperiment::colData(data)).

group1 — A group from group.col. ELMER will run group1 vs group2. That means, if direction is hyper, get probes hypermethylated in group 1 compared to group 2.

group2 — A group from group.col. ELMER will run group1 vs group2. That means, if direction is hyper, get probes hypermethylated in group 1 compared to group 2.

cores — A interger which defines number of core to be used in parallel process. Default is 1: don't use parallel process.

correlation — Type of correlation to evaluate (negative or positive). Negative (default) checks if hypomethylated region has a upregulated target gene. Positive checks if region hypermethylated has a upregulated target gene.

filter.probes — Should filter probes by selecting only probes that have at least a certain number of samples below and above a certain cut-off. See preAssociationProbeFiltering function.

filter.portion — A number specify the cut point to define binary methylation level for probe loci. Default is 0.3. When beta value is above 0.3, the probe is methylated and vice versa. For one probe, the percentage of methylated and unmethylated samples should be above filter.percentage value. Only used if filter.probes is TRUE. See preAssociationProbeFiltering function.

filter.percentage — Minimun percentage of samples to be considered in methylated and unmethylated for the filter.portion option. Default 5%. Only used if filter.probes is TRUE. See preAssociationProbeFiltering function.

label — A character labels the outputs.

addDistNearestTSS — Calculated distance to the nearest TSS instead of gene distance. Having to calculate the distance to nearest TSS will take some time.

save — Two files will be saved if save is true: getPair.XX.all.pairs.statistic.csv and getPair.XX.pairs.significant.csv (see detail).

Value

Statistics for all pairs and significant pairs

Author(s)

Lijing Yao (creator: lijingya@usc.edu) Tiago C Silva (maintainer: tiagochst@usp.br)

References

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

Examples

```
data <- ELMER:::getdata("elmer.data.example")
nearGenes <- GetNearGenes(TRange=getMet(data)[c("cg00329272","cg10097755"),],
                          geneAnnot=getExp(data))
Hypo.pair <- get.pair(data=data,
                      nearGenes=nearGenes,
                      permu.size=5,
                      group.col = "definition",
                      group1 = "Primary solid Tumor",
                      group2 = "Solid Tissue Normal",
                      raw.pvalue = 0.2,
                      Pe = 0.2,
                      dir.out="./",
                      label= "hypo")

Hypo.pair <- get.pair(data = data,
                      nearGenes = nearGenes,
                      permu.size = 5,
                      raw.pvalue = 0.2,
                      Pe = 0.2,
                      dir.out = "./",
                      diffExp = TRUE,
                      group.col = "definition",
                      group1 = "Primary solid Tumor",
                      group2 = "Solid Tissue Normal",
                      label = "hypo")
```

---

get.permu — *get.permu to generate permutation results for calculation of empirical P values for each enhancer-gene linkage.*

Description

get.permu is a function to use the same statistic model to calculate random enhancer-gene pairs. Based on the permutation value, empirical P value can be calculated for the real enhancer-gene pair (see reference).

Usage

```
get.permu(data,
          geneID,
          methy = NULL,
          unmethy = NULL,
          percentage = 0.2,
          rm.probes = NULL,
          correlation = "negative",
          permu.size = 10000,
          permu.dir = NULL,
          cores = 1)
```

Arguments

data	A multiAssayExperiment with DNA methylation and Gene Expression data. See createMAE function.
geneID	A vector lists the genes' ID.
methy	Index of M (methylated) group.
unmethy	Index of U (unmethylated) group.
percentage	A number ranges from 0 to 1 specifying the percentage of samples of group 1 and group 2 groups used to link probes to genes. Default is 0.2.
rm.probes	A vector lists the probes name.
correlation	Type of correlation to identify. Default is negative: look for hypomethylation and increase target expression.
permu.size	A number specify the times of permuation. Default is 10000.
permu.dir	A path where the output of permuation will be.
cores	A interger which defines number of core to be used in parallel process. Default is 1: don't use parallel process.

Value

Permutations

Note

Permutation is the most time consuming step. It is recommended to use multiple cores for this step. Default permutation time is 1000 which may need 12 hrs by 4 cores. However 10,000 permutations is recommended to get high confidence results. But it may cost 2 days.

Author(s)

Lijing Yao (creator: lijingya@usc.edu) Tiago C Silva (maintainer: tiagochst@usp.br)

References

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

Examples

```
data <- ELMER:::getdata("elmer.data.example")
permu <-get.permu(data = data,
                  geneID=rownames(getExp(data)),
                  rm.probes=c("cg00329272","cg10097755"),
                  permu.size=5)
```

`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.

`get.tab` *summarize MR TF as a binary table with 1 if TF was found in the analysis, 0 if not*

Description

summarize MR TF as a binary table with 1 if TF was found in the analysis, 0 if not

Usage

```
get.tab(dir, classification, top = TRUE)
```

Arguments

`dir`	Directory with ELMER results
`classification`	Which columns to retrieve family or subfamily
`top`	Consider only top 1 within each (sub)family

Examples

```
## Not run:
dir.create("out")
dir.create("out2")
data <- tryCatch(
  ELMER:::getdata("elmer.data.example"),
  error = function(e) {
    message(e)
     data(elmer.data.example, envir = environment())
  })
enriched.motif <- list("P53_HUMAN.H11MO.1.A"= c("cg00329272", "cg10097755", "cg08928189",
                                "cg17153775", "cg21156590", "cg19749688", "cg12590404",
                                "cg24517858", "cg00329272", "cg09010107", "cg15386853",
                                "cg10097755", "cg09247779", "cg09181054"))
TF <- get.TFs(data,
              enriched.motif,
              group.col = "definition",
              group1 = "Primary solid Tumor",
              group2 = "Solid Tissue Normal",
              TFs = data.frame(
                     external_gene_name=c("TP53","TP63","TP73"),
                     ensembl_gene_id= c("ENSG00000141510",
                                        "ENSG00000073282",
                                        "ENSG00000078900"),
                     stringsAsFactors = FALSE),
                     dir.out = "out",
             label="hypo")
TF <- get.TFs(data,
              enriched.motif,
              group.col = "definition",
              group1 = "Primary solid Tumor",
              group2 = "Solid Tissue Normal",
              TFs = data.frame(
                     external_gene_name=c("TP53","TP63","TP73"),
                     ensembl_gene_id= c("ENSG00000141510",
                                        "ENSG00000073282",
                                        "ENSG00000078900"),
                     stringsAsFactors = FALSE),
                     dir.out = "out2",
             label="hypo")
 ta.family <- get.tab(dir = c("out","out2"),classification = "family")
 ta.subfamily <- get.tab(dir = c("out","out2"),classification = "subfamily")
 unlink("out")
 unlink("out2")

## End(Not run)
```

`get.tabs` *Creating matrix for MR TF heatmap*

Description

Code used to create matrix for MR TF heatmap

Usage

```
get.tabs(dir, classification = "family", top = TRUE)
```

Arguments

dir	Vector ofr directory with results
classification	Consider family or subfamily
top	Consider only top 1 within each (sub)family

Examples

```
## Not run:
elmer.results <- dirname(
dir(path = "analysis",
   pattern = "*.hypo.pairs.significant.csv",
   recursive = T,
   full.names = T,
   all.files = T))
tabs <- get.tabs(dir = elmer.results, classification = "subfamily")

## End(Not run)
```

---

get.TFs *get.TFs to identify regulatory TFs.*

Description

get.TFs is a function to identify regulatory TFs based on motif analysis and association analysis between the probes containing a particular motif and expression of all known TFs. If save is true, two files will be saved: getTF.XX.significant.TFs.with.motif.summary.csv and getTF.hypo.TFs.with.motif.pvalue.rda (see detail).

Usage

```
get.TFs(data,
          enriched.motif,
          TFs,
          group.col,
          group1,
          group2,
          mode = "unsupervised",
          correlation = "negative",
          diff.dir = NULL,
          motif.relevant.TFs,
          minSubgroupFrac = 0.4,
          dir.out = "./",
          label = NULL,
          save.plots = FALSE,
          cores = 1,
          topTFper = 0.05,
          save = TRUE)
```

Arguments

data	A multiAssayExperiment with DNA methylation and Gene Expression data. See createMAE function.
enriched.motif	A list containing output of get.enriched.motif function or a path of XX.rda file containing output of get.enriched.motif function.
TFs	A data.frame containing TF GeneID and Symbol or a path of XX.csv file containing TF GeneID and Symbol. If missing, human.TF list will be used (human.TF data in ELMER.data). For detail information, refer the reference paper.
group.col	A column defining the groups of the sample. You can view the available columns using: colnames(MultiAssayExperiment::colData(data)).
group1	A group from group.col. ELMER will run group1 vs group2. That means, if direction is hyper, get probes hypermethylated in group 1 compared to group 2.
group2	A group from group.col. ELMER will run group1 vs group2. That means, if direction is hyper, get probes hypermethylated in group 1 compared to group 2.
mode	A character. Can be "unsupervised" or "supervised". If unsupervised is set the U (unmethylated) and M (methylated) groups will be selected among all samples based on methylation of each probe. Otherwise U group and M group will set as the samples of group1 or group2 as described below: If diff.dir is "hypo, U will be the group 1 and M the group2. If diff.dir is "hyper" M group will be the group1 and U the group2.
correlation	Type of correlation to evaluate (negative or positive). Negative checks if hypomethylated is upregulated. Positive if hypermethylated is upregulated.
diff.dir	A character can be "hypo" or "hyper", showing differential methylation direction in group 1. It can be "hypo" which means the probes are hypomethylated in group1; "hyper" which means the probes are hypermethylated in group1; This argument is used only when mode is supervised nad it should be the same value from get.diff.meth function.
motif.relevant.TFs	A list containing motif as names and relavent TFs as contents for each list element or a path of XX.rda file containing a list as above. If missing, motif.relavent.TFs will be used (motif.relavent.TFs data in ELMER.data). For detail information, refer the reference paper.
minSubgroupFrac	A number ranging from 0 to 1 specifying the percentage of samples used to create the groups U (unmethylated) and M (methylated) used to link probes to TF expression. Default is 0.4 (lowest quintile of all samples will be in the U group and the highest quintile of all samples in the M group).
dir.out	A path specifies the directory for outputs of get.pair function. Default is current directory
label	A character labels the outputs.
save.plots	Create TF ranking plots ?
cores	A interger which defines the number of cores to be used in parallel process. Default is 1: no parallel process.
topTFper	Top ranked TF to be retrieved (default "0.05" - 5 percent)
save	A logic. If save is ture, two files will be saved: getTF.XX.significant.TFs.with.motif.summary.csv and getTF.hypo.TFs.with.motif.pvalue.rda (see detail). If save is false, a data frame contains the same content with the first file.

Details

save: If save is ture, two files will be saved. The first file is getTF.XX.significant.TFs.with.motif.summary.csv (XX depends on option lable). This file contain the regulatory TF significantly associate with average DNA methylation at particular motif sites. The second file is getTF.hypo.TFs.with.motif.pvalue.rda (XX depends on option label). This file contains a matrix storing the statistic results for significant associations between TFs (row) and average DNA methylation at motifs (column). If save is false, a data frame which contains the same content with the first file will be reported.

Value

Potential responsible TFs will be reported in a dataframe with 4 columns:

- motif: the names of motif.
- top.potential.TF.family: the highest ranking upstream TFs which are known recognized the motif. First item in potential.TFs.family
- top.potential.TF.subfamily: the highest ranking upstream TFs which are known recognized the motif. First item in potential.TFs.subfamily
- potential.TFs.family: TFs which are within top 5% list and are known recognized the motif (considering family classification).
- potential.TFs.subfamily: TFs which are within top 5% list and are known recognized the motif (considering subfamily classification).
- top_5percent: all TFs which are within top 5% list.

Author(s)

Lijing Yao (creator: lijingya@usc.edu) Tiago C Silva (maintainer: tiagochst@usp.br)

References

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

Examples

```
data <- tryCatch(
  ELMER:::getdata("elmer.data.example"),
  error = function(e) {
    message(e)
     data(elmer.data.example, envir = environment())
  })
enriched.motif <- list(
   "P53_HUMAN.H11MO.1.A"= c(
      "cg00329272", "cg10097755", "cg08928189",
      "cg17153775", "cg21156590", "cg19749688", "cg12590404",
      "cg24517858", "cg00329272", "cg09010107", "cg15386853",
      "cg10097755", "cg09247779", "cg09181054"
      )
)
TF <- get.TFs(
   data,
   enriched.motif,
   group.col = "definition",
   group1 = "Primary solid Tumor",
```

```
   group2 = "Solid Tissue Normal",
   TFs = data.frame(
         external_gene_name=c("TP53","TP63","TP73"),
         ensembl_gene_id= c(
           "ENSG00000141510",
           "ENSG00000073282",
           "ENSG00000078900"
           ),
           stringsAsFactors = FALSE
           ),
 label = "hypo"
)
# This case will use Uniprot dabase to get list of Trasncription factors
TF <- get.TFs(
  data,
  group.col = "definition",
  group1 = "Primary solid Tumor",
  group2 = "Solid Tissue Normal",
  enriched.motif,
  label = "hypo"
)
```

`get450K` *get450K to download HM40K DNA methylation data for certain cancer types from TCGA website. @description get450K is a function to download latest version of HM450K DNA methylation for all samples of certain cancer types from GDC website.*

Description

get450K to download HM40K DNA methylation data for certain cancer types from TCGA website. @description get450K is a function to download latest version of HM450K DNA methylation for all samples of certain cancer types from GDC website.

Usage

```
get450K(disease, basedir="./Data",filter=0.2, genome = "hg38")
```

Arguments

`disease`	A character specifies the disease to download from TCGA such as BLCA
`basedir`	A path. Shows where the data will be stored.
`filter`	For each probe, the percentage of NA among the all the samples should smaller than filter.
`genome`	Data aligned against which genome of reference. Options: "hg38" (default)

Value

Download all DNA methylation from HM450K level 3 data for the specified disease.

`getClinic` *getClinic to download clinic data for certain cancer types from TCGA website.*

Description

getClinic is a function to download latest version of clinic data for all samples of certain cancer types from TCGA website.

Usage

```
getClinic(disease, basedir = "./Data")
```

Arguments

`disease`	A character specifies the disease to download from TCGA such as BLCA
`basedir`	A path shows where the data will be stored.

Value

Download all clinic information for the specified disease.

`getExp` *Get Gene expression object from MAE*

Description

Get Gene expression object from MAE

Usage

```
getExp(data)
```

Arguments

`data`	A multiAssayExperiment with DNA methylation and Gene Expression data. See `createMAE` function.

getExpSamples *Get Gene expression object samples from MAE*

Description

Get Gene expression object samples from MAE

Usage

```
getExpSamples(data)
```

Arguments

data	A multiAssayExperiment with DNA methylation and Gene Expression data. See createMAE function.

getGeneID *getGeneID to report gene id from symbol*

Description

getGeneID to report gene id from symbol

Usage

```
getGeneID(data, symbol)
```

Arguments

data	A multiAssayExperiment with DNA methylation and Gene Expression data. See createMAE function.
symbol	A vector of characters which are gene symbols

Value

The gene ID for these gene symbols

Examples

```
data <- ELMER:::getdata("elmer.data.example")
getGeneID(data, symbol="ZNF697")
```

getMet — *Get DNA methylation object from MAE*

Description

Get DNA methylation object from MAE

Usage

```
getMet(data)
```

Arguments

data	A multiAssayExperiment with DNA methylation and Gene Expression data. See createMAE function.

getMetSamples — *Get DNA methylation object samples from MAE*

Description

Get DNA methylation object samples from MAE

Usage

```
getMetSamples(data)
```

Arguments

data	A multiAssayExperiment with DNA methylation and Gene Expression data. See createMAE function.

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.

Examples

```
geneAnnot <- getTSS(genome = "hg38")
probe <- GenomicRanges::GRanges(seqnames = c("chr1","chr2"),
range=IRanges::IRanges(start = c(16058489,236417627), end= c(16058489,236417627)),
name= c("cg18108049","cg17125141"))
names(probe) <- c("cg18108049","cg17125141")
NearbyGenes <- GetNearGenes(numFlankingGenes = 20,geneAnnot=geneAnnot,TRange=probe)
```

---

getRandomPairs *Get random pairs*

Description

This function will receive a pair gene probes and will return a random object with the following pattern, if a probe is linked to R1 and L3 genes the random pairs will be a random probes (a distal probe not in the input pairs) also linked to its R1 and L3 gene.

Usage

```
getRandomPairs(pairs, genome = "hg38", met.platform = "450K", cores = 1)
```

Arguments

pairs	A data frame with probe, gene and side information. See example below.
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	A interger which defines the number of cores to be used in parallel process. Default is 1: no parallel process.

Value

A data frame with the random linkages

Examples

```
## Not run:
 data <- ELMER:::getdata("elmer.data.example")
 nearGenes <- GetNearGenes(TRange=getMet(data)[c("cg00329272","cg10097755"),],
                           geneAnnot=getExp(data))

 pair <- get.pair(data = data,
                  group.col = "definition",
                  group1 = "Primary solid Tumor",
                  group2 = "Solid Tissue Normal",
                  mode = "supervised",
                  diff.dir = "hypo",
                  nearGenes = nearGenes,
                  permu.size = 5,
                  raw.pvalue =  0.001,
                  Pe = 0.2,
                  dir.out="./",
                  permu.dir = "permu_test",
                  label = "hypo")

## End(Not run)
 pair <- data.frame(Probe = rep("cg00329272",3),
                    GeneID = c("ENSG00000116213","ENSG00000130762","ENSG00000149527"),
                    Sides = c("R5","R2","L4"))
 getRandomPairs(pair)
```

---

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)

Examples

```
geneAnnot <-  ELMER:::get.GRCh("hg38",as.granges = TRUE)
tssAnnot <-  getTSS(genome = "hg38")
probe <- GenomicRanges::GRanges(seqnames = c("chr1","chr2"),
range=IRanges::IRanges(start = c(16058489,236417627), end= c(16058489,236417627)),
name= c("cg18108049","cg17125141"))
names(probe) <- c("cg18108049","cg17125141")
NearbyGenes <- getRegionNearGenes(numFlankingGenes = 20,
                            geneAnnot = geneAnnot,
                            TRange = probe,
                            tssAnnot = tssAnnot)
```

---

getRNAseq *getRNAseq to download all RNAseq data for a certain cancer type from TCGA.*

Description

getRNAseq is a function to download RNAseq data for all samples of a certain cancer type from TCGA

Usage

```
getRNAseq(disease, basedir = "./Data", genome = "hg38")
```

Arguments

disease	A character specifies disease in TCGA such as BLCA
basedir	Download all RNA seq level 3 data for the specified disease.
genome	Data aligned against which genome of reference. Options: "hg38" (default)

Value

Download all RNA seq level 3 data for the specified disease.

---

getSymbol *getSymbol to report gene symbol from id*

Description

getSymbol to report gene symbol from id

Usage

```
getSymbol(data, geneID)
```

Arguments

data	A multiAssayExperiment with DNA methylation and Gene Expression data. See createMAE function.
geneID	A character which is the ensembl_gene_id

Value

The gene symbol for input genes.

Examples

```
data <- ELMER:::getdata("elmer.data.example")
getSymbol(data, geneID="ENSG00000143067")
```

---

getTCGA *getTCGA to download DNA methylation, RNA expression and clinic data for all samples of certain cancer type from TCGA.*

Description

getTCGA is a function to download DNA methylation, RNA expression and clinic data for all samples of certain cancer type from TCGA website. And downloaded data will be transform to matrixes or data frame for further analysis.

Usage

```
getTCGA(disease, Meth=TRUE, RNA=TRUE, Clinic=TRUE, basedir="./Data", genome = "hg38")
```

Arguments

disease	A character specifies the disease to download in TCGA such as BLCA
Meth	A logic if TRUE HM450K DNA methylation data will download.
RNA	A logic if TRUE RNA-seq Hiseq-V2 from TCGA level 3 will be download.
Clinic	A logic if TRUE clinic data will be download for that disease.
basedir	A path shows where the data will be stored.
genome	Data aligned against which genome of reference. Options: "hg38" (default)

Value

Download DNA methylation (HM450K)/RNAseq(HiseqV2)/Clinic data for the specified disease from TCGA.

Examples

```
getTCGA(
   disease = "BRCA",
   Meth = FALSE,
   RNA = FALSE,
   Clinic = TRUE,
   basedir = tempdir(),
   genome = "hg38"
)
```

---

getTF *Get human TF list from the UNiprot database*

Description

This function gets the last version of human TF list from the UNiprot database

Usage

```
getTF()
```

Value

A data frame with the ensemble gene id.

getTFBindingSites *Get MR TF binding regions infered by ELMER*

Description

Saves a bed file with the unmethylated probes (+-250bp) regions that was infered to be bound by a given TF

Usage

```
getTFBindingSites(
  tf = NULL,
  results.dir = NULL,
  genome = "hg38",
  met.platform = "450K"
)
```

Arguments

tf	TF name
results.dir	path to the directory with the results (i.e. analysis/unsupervised/definition-Primary.solid.Tumor_vs_S
genome	Human genome (hg38, hg19)
met.platform	DNA Methylation Array platform (EPIC, 450K)

Examples

```
## Not run:
  getTFBindingSites("HNF1A",
                    results.dir = "analysis/unsupervised/group-Tumor_vs_Normal/hypo/")

## End(Not run)
```

getTFtargets *Get TF target genes*

Description

This function uses ELMER analysis results and summarizes the possible genes targets for each TF

Usage

```
getTFtargets(
  pairs,
  enriched.motif,
  TF.result,
  dmc.analysis,
  mae,
  save = TRUE,
  dir.out = "./",
```

```
  classification = "family",
  cores = 1,
  label = NULL
)
```

Arguments

pairs	Output of get.pairs function: dataframe or file path
enriched.motif	List of probes for each enriched motif: list of file path. The file created by ELMER is getMotif...enriched.motifs.rda
TF.result	Output get.TF function: dataframe or file path
dmc.analysis	DMC results file or data frame
mae	A multiAssayExperiment outputed from createMAE function
save	A logic. If save is true, a files will be saved: getTFtarget.XX..csv If save is false, only a data frame contains the same content with the first file.
dir.out	A path specifies the directory for outputs of get.pair function. Default is current directory
classification	use family or subfamily classification to consider potential TF
cores	Number of cores to be used in parallel
label	A character labels the outputs.

Examples

```
pairs <- data.frame(Probe = c("cg26992600","cg26992800","cg26992900"),
                    Symbol = c("KEAP1","DSP","ATP86"))
enriched.motif <- list("FOXD3_HUMAN.H11MO.0.D"= c("cg26992800","cg26992900"))
TF.result <- data.frame(motif = c("FOXD3_HUMAN.H11MO.0.D"),
                        potential.TF.family = c("TP63;TP73"))
getTFtargets(pairs,enriched.motif,TF.result)

## Not run:
getTFtargets("../LUAD_LUSC_analysis_hg38/hyper/getPair.hyper.pairs.significant.csv",
enriched.motif = "../LUAD_analysis_hg38/hyper/getMotif.hyper.enriched.motifs.rda",
TF.result = "../LUAD_analysis_hg38/hyper/getTF.hyper.significant.TFs.with.motif.summary.csv")

## End(Not run)
```

---

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)

Examples

```
# get GENCODE gene annotation (transcripts level)
## Not run:
    getTSS <- getTSS()
    getTSS <- getTSS(genome.build = "hg38", TSS=list(upstream=1000, downstream=1000))

## End(Not run)
```

---

heatmapGene *Heatmap for correlation between probes DNA methylation and a single gene expression.*

Description

This heatmap will sort samples by their gene expression and show the DNA methylation levels of the paired probes to that gene. If no pairs are given, nearest probes will be selected. To use this function you MAE object (input data) will need all probes and not only the distal ones. This plot can be used to evaluate promoter, and intro, exons regions and closer distal probes of a gene to verify if their DNA methylation level is affecting the gene expression

Usage

```
heatmapGene(
  data,
  group.col,
  group1,
  group2,
  pairs,
  GeneSymbol,
  scatter.plot = FALSE,
  correlation.method = "pearson",
```

```
  correlation.table = FALSE,
  annotation.col = NULL,
  met.metadata = NULL,
  exp.metadata = NULL,
  dir.out = ".",
  filter.by.probe.annotation = TRUE,
  numFlankingGenes = 10,
  width = 10,
  height = 10,
  scatter.plot.width = 10,
  scatter.plot.height = 10,
  filename = NULL
)
```

Arguments

data	A MultiAssayExperiment with a DNA methylation SummarizedExperiment (all probes) and a gene Expression SummarizedExperiment.
group.col	A column from the sample matrix from the MultiAssayExperiment object. Accessed with colData(mae)
group1	A group from group.col. ELMER will run group1 vs group2. That means, if direction is hyper, get probes hypermethylated in group 1 compared to group 2.
group2	A group from group.col. ELMER will run group1 vs group2. That means, if direction is hyper, get probes hypermethylated in group 1 compared to group 2.
pairs	List of probe and pair genes
GeneSymbol	Gene Symbol
scatter.plot	Plot scatter plots
correlation.method	Correlation method: Pearson or sperman
correlation.table	save table with spearman correlation analysis ?
annotation.col	A vector of columns from the sample matrix from the MultiAssayExperiment object. Accessed with colData(mae) to be added as annotation to the heatmap
met.metadata	A vector of metdatada columns available in the DNA methylation GRanges to should be added to the heatmap.
exp.metadata	A vector of metdatada columns available in the Gene expression GRanges to should be added to the heatmap.
dir.out	Where to save the plots
filter.by.probe.annotation	Filter probes to plot based on probes annotation
numFlankingGenes	numFlankingGenes to plot.
width	Figure width
height	Figure height
scatter.plot.width	Scatter plot width
scatter.plot.height	Scatter plot height
filename	File names (.pdf) to save the file (i.e. "plot.pdf"). If NULL return plot.

Value

A heatmap

Author(s)

Tiago Chedraoui Silva (tiagochst at gmail.com)

Examples

```
## Not run:
  data <- ELMER:::getdata("elmer.data.example")
  group.col <- "subtype_Expression.Subtype"
  group1 <- "classical"
  group2 <- "secretory"
  pairs <- data.frame(ID = c("cg15924102","cg19403323", "cg22396959"),
                   GeneID = c("ENSG00000196878", "ENSG00000009790", "ENSG00000009790" ),
                      Symbol = c("TRAF3IP3","LAMB3","LAMB3"),
                      Side = c("R1","L1","R3"),
                      Distance = c(6017,168499,0),
                      stringsAsFactors = FALSE)
 heatmapGene(data = data,
             group.col = group.col,
             group1 = group1,
             group2 = group2,
             pairs = pairs,
             GeneSymbol = "LAMB3",
             height = 5,
             annotation.col = c("ethnicity","vital_status"),
             filename = "heatmap.pdf")
 \dontrun{
     heatmapGene(data = data,
                 group.col = group.col,
                 group1 = group1,
                 group2 = group2,
                 GeneSymbol = "ACP6",
                 annotation.col = c("ethnicity","vital_status"),
                 filename = "heatmap_closer_probes.pdf")
 }

## End(Not run)
```

`heatmapPairs` *Heatmap of pairs gene and probes anti-correlated*

Description

Heatmp plot of pairs gene and probes anti-correlated

Usage

```
heatmapPairs(
  data,
  group.col,
```

```
  group1,
  group2,
  pairs,
  subset = FALSE,
  cluster.within.groups = TRUE,
  plot.distNearestTSS = FALSE,
  annotation.col = NULL,
  met.metadata = NULL,
  exp.metadata = NULL,
  width = 10,
  height = 7,
  filename = NULL
)
```

Arguments

data	A MultiAssayExperiment with a DNA methylation SummarizedExperiment (all probes) and a gene Expression SummarizedExperiment.
group.col	A column from the sample matrix from the MultiAssayExperiment object. Accessed with colData(mae)
group1	A group from group.col. ELMER will run group1 vs group2. That means, if direction is hyper, get probes hypermethylated in group 1 compared to group 2.
group2	A group from group.col. ELMER will run group1 vs group2. That means, if direction is hyper, get probes hypermethylated in group 1 compared to group 2.
pairs	List of probe and pair genes
subset	Subset MAE object to keep only groups compared ?
cluster.within.groups	Cluster columns based on the groups
plot.distNearestTSS	Plot track with distNearestTSS ?
annotation.col	A vector of columns from the sample matrix from the MultiAssayExperiment object. Accessed with colData(mae) to be added as annotation to the heatmap.
met.metadata	A vector of metdatada columns available in the DNA methylation GRanges to should be added to the heatmap.
exp.metadata	A vector of metdatada columns available in the Gene expression GRanges to should be added to the heatmap.
width	Figure width
height	Figure height
filename	File names (.pdf) to save the file (i.e. "plot.pdf"). If NULL return plot.

Value

A heatmap

Author(s)

Tiago Chedraoui Silva (tiagochst at gmail.com)

Examples

```
## Not run:
  data <- ELMER:::getdata("elmer.data.example")
  group.col <- "subtype_Expression.Subtype"
  group1 <- "classical"
  group2 <- "secretory"
  pairs <- data.frame(Probe = c("cg15924102","cg19403323", "cg22396959"),
                      GeneID = c("ENSG00000196878", "ENSG00000009790", "ENSG00000009790" ),
                        Symbol = c("TRAF3IP3","LAMB3","LAMB3"),
                        Distance = c(6017,168499,0),
                        Raw.p = c(0.001,0.00001,0.001),
                        Pe = c(0.001,0.00001,0.001))
 heatmapPairs(
   data = data, group.col = group.col,
              group1 = group1, group2 = group2,
              annotation.col = c("ethnicity","vital_status","age_at_diagnosis"),
              pairs, filename = "heatmap.pdf",
              height = 4, width = 11
  )

## End(Not run)
```

`lm_eqn` *lable linear regression formula*

Description

lable linear regression formula

Usage

```
lm_eqn(df, Dep, Exp)
```

Arguments

`df`	A data.frame object contains two variables: dependent variable (Dep) and explanation variable (Exp).
`Dep`	A character specify dependent variable. The first column will be dependent variable as default.
`Exp`	A character specify explanation variable. The second column will be explanation variable as default.

Value

A linear regression formula

`metBoxPlot` *scatter.plot to plot scatter plots between gene expression and DNA methylation.*

Description

scatter.plot is a function to plot various scatter plots between gene expression and DNA methylation. When byPair is specified, scatter plot for individual probe-gene pairs will be generated. When byProbe is specified, scatter plots for one probes with nearby 20 gene pairs will be generated. When byTF is specified, scatter plot for TF expression and average DNA methylation at certain motif sites will be generated.

Usage

```
metBoxPlot(
  data,
  group.col,
  group1,
  group2,
  probe,
  min.samples = 5,
  minSubgroupFrac = 0.2,
  diff.dir = "hypo",
  legend.col = NULL,
  title = NULL,
  filename = NULL,
  save = TRUE
)
```

Arguments

`data`	A multiAssayExperiment with DNA methylation and Gene Expression data. See `createMAE` function.
`group.col`	A column defining the groups of the sample. You can view the available columns using: colnames(MultiAssayExperiment::colData(data)).
`group1`	A group from group.col. ELMER will run group1 vs group2. That means, if direction is hyper, get probes hypermethylated in group 1 compared to group 2.
`group2`	A group from group.col. ELMER will run group1 vs group2. That means, if direction is hyper, get probes hypermethylated in group 1 compared to group 2.
`probe`	Character with probe name (i.e. "cg24517858")
`min.samples`	Minimun number of samples to use in the analysis. Default 5. If you have 10 samples in one group, percentage is 0.2 this will give 2 samples in the lower quintile, but then 5 will be used.
`minSubgroupFrac`	A number ranges from 0 to 1 specifying the percentage of samples from group1 and group2 that are used to identify the differential methylation. Default is 0.2 because we did not expect all cases to be from a single molecular subtype.But, If you are working with molecular subtypes please set it to 1.

diff.dir	A character can be "hypo" or "hyper", showing differential methylation dirction. It can be "hypo" which is only selecting hypomethylated probes; "hyper" which is only selecting hypermethylated probes;
legend.col	legend title
title	plot title
filename	File names (.png) to save the file (i.e. "plot.png")
save	Save plot as PNG

Value

Box plot

Author(s)

Tiago Chedraoui Silva (tiagochst at gmail.com)

Examples

```
## Not run:
  data <- ELMER:::getdata("elmer.data.example")
  group.col <- "subtype_Expression.Subtype"
  group1 <- "classical"
  group2 <- "secretory"
  metBoxPlot(data,
             group.col = group.col,
             group1 = group1,
             group2 = group2,
             probe ="cg17898069",
             minSubgroupFrac = 0.2,
             diff.dir = "hypo")

## End(Not run)
```

---

motif.enrichment.plot *motif.enrichment.plot to plot bar plots showing motif enrichment ORs and 95% confidence interval for ORs*

Description

motif.enrichment.plot to plot bar plots showing motif enrichment ORs and 95% confidence interval for ORs. Option motif.enrichment can be a data frame generated by get.enriched.motif or a path of XX.csv saved by the same function.

Usage

```
motif.enrichment.plot(motif.enrichment,
                      significant = NULL,
                      dir.out ="./",
                      save = TRUE,
                      label = NULL,
                      title = NULL,
                      width = 10,
                      height = NULL,
                      summary = FALSE)
```

Arguments

motif.enrichment	A data frame or a file path of get.enriched.motif output motif.enrichment.csv file.
significant	A list to select subset of motif. Default is NULL.
dir.out	A path specify the directory to which the figures will be saved. Current directory is default.
save	A logic. If true (default), figure will be saved to dir.out.
label	A character. Labels the outputs figure.
title	Plot title. Default: no title
width	Plot width
height	Plot height. If NULL a default value will be calculated
summary	Create a summary table along with the plot, it is necessary to add two new columns to object (NumOfProbes and PercentageOfProbes)

Details

motif.enrichment If input data.frame object, it should contain "motif", "OR", "lowerOR", "upperOR" columns. motif specifies name of motif; OR specifies Odds Ratio, lowerOR specifies lower boundary of OR (95 upperOR specifies upper boundary of OR(95

significant A list used to select subset of motif.enrichment by the cutoff of OR, lowerOR, upperOR. significant=list(OR=1). More than one cutoff can be specified such as significant = list(OR=1, lowerOR=1,upperOR=4)

Value

A figure shows the enrichment level for selected motifs.

Author(s)

Lijing Yao (creator: lijingya@usc.edu)

References

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

Examples

```
motif.enrichment <- data.frame(motif = c("TP53","NR3C1","E2F1","EBF1","RFX5","ZNF143", "CTCF"),
                               OR = c(19.33,4.83,1, 4.18, 3.67,3.03,2.49),
                               lowerOR = c(10,3,1.09,1.9,1.5,1.9, 0.82),
                               upperOR = c(23,5,3,7,6,5,5),
                               stringsAsFactors = FALSE)
motif.enrichment.plot(motif.enrichment = motif.enrichment,
                      significant = list(OR = 3),
                      label = "hypo", save = FALSE)
motif.enrichment.plot(motif.enrichment = motif.enrichment,
                      significant = list(OR = 3),
                      label = "hypo",
                      title = "OR for paired probes hypomethylated in Mutant vs WT",
```

```
                        save = FALSE)
motif.enrichment <- data.frame(motif = c("TP53","NR3C1","E2F1","EBF1","RFX5","ZNF143", "CTCF"),
                               OR = c(19.33,4.83,1, 4.18, 3.67,3.03,2.49),
                               lowerOR = c(10,3,1.09,1.9,1.5,1.5, 0.82),
                               upperOR = c(23,5,3,7,6,5,5),
                               NumOfProbes = c(23,5,3,7,6,5,5),
                              PercentageOfProbes = c(0.23,0.05,0.03,0.07,0.06,0.05,0.05),
                               stringsAsFactors=FALSE)
motif.enrichment.plot(motif.enrichment = motif.enrichment,
                      significant = list(OR = 3),
                      label = "hypo", save = FALSE)
motif.enrichment.plot(motif.enrichment = motif.enrichment,
                      significant = list(OR = 3),
                      label = "hypo",
                      summary = TRUE,
                      title = "OR for paired probes hypomethylated in Mutant vs WT",
                      save = TRUE)
```

`preAssociationProbeFiltering` *Filtering probes*

Description

This function has some filters to the DNA methylation data in each it selects probes to avoid correlations due to non-cancer contamination and for additional stringency.

- Filter 1: We usually call locus unmethylated when the methylation value < 0.3 and methylated when the methylation value > 0.3. Therefore Meth_B is the percentage of methylation value > K. Basically, this step will make sure we have at least a percentage of beta values lesser than K and n percentage of beta values greater K. For example, if percentage is 5%, the number of samples 100 and K = 0.3, this filter will select probes that we have at least 5 (5% of 100%) samples have beta values > 0.3 and at least 5 samples have beta values < 0.3. This filter is importante as true promoters and enhancers usually have a pretty low value (of course purity can screw that up). we often see lots of PMD probes across the genome with intermediate values like 0.4. Choosing a value of 0.3 will certainly give some false negatives, but not compared to the number of false positives we thought we might get without this filter.

Usage

```
preAssociationProbeFiltering(data, K = 0.3, percentage = 0.05)
```

Arguments

`data`	A MultiAssayExperiment with a DNA methylation martrix or a DNA methylation matrix
`K`	Cut off to consider probes as methylated or unmethylated. Default: 0.3
`percentage`	The percentage of samples we should have at least considered as methylated and unmethylated

Value

An object with the same class, but with the probes removed.

References

Yao, Lijing, et al. "Inferring regulatory element landscapes and transcription factor networks from cancer methylomes." Genome biology 16.1 (2015): 1. Method section (Linking enhancer probes with methylation changes to target genes with expression changes).

Examples

```
random.probe <- runif(100, 0, 1)
bias_l.probe <- runif(100, 0, 0.3)
bias_g.probe <- runif(100, 0.3, 1)
met <- rbind(random.probe,bias_l.probe,bias_g.probe)
met <- preAssociationProbeFiltering(data = met,  K = 0.3, percentage = 0.05)
met <- rbind(random.probe,random.probe,random.probe)
met <- preAssociationProbeFiltering(met,  K = 0.3, percentage = 0.05)
data <- ELMER:::getdata("elmer.data.example") # Get data from ELMER.data
data <- preAssociationProbeFiltering(data,  K = 0.3, percentage = 0.05)

cg24741609 <- runif(100, 0, 1)
cg17468663 <- runif(100, 0, 0.3)
cg14036402 <- runif(100, 0.3, 1)
met <- rbind(cg24741609,cg14036402,cg17468663)
colnames(met) <- paste("sample",1:100)
exp <- met
rownames(exp) <- c("ENSG00000141510","ENSG00000171862","ENSG00000171863")
sample.info <- S4Vectors::DataFrame(primary = paste("sample",1:100),
                                    sample.type = rep(c("Normal", "Tumor"),50))
rownames(sample.info) <- colnames(exp)
mae <- createMAE(exp = exp, met = met, colData = sample.info, genome = "hg38")
mae <- preAssociationProbeFiltering(mae,  K = 0.3, percentage = 0.05)
```

`promoterMeth` *promoterMeth to calculate associations of gene expression with DNA methylation at promoter regions*

Description

promoterMeth is a function to calculate associations of gene expression with DNA methylation at promoter regions.

Usage

```
promoterMeth(data, sig.pvalue = 0.01, minSubgroupFrac = 0.4,
             upstream = 200,  downstream = 2000, save = TRUE, cores = 1)
```

Arguments

`data`	A Multi Assay Experiment object with DNA methylation and gene expression Summarized Experiment objects
`sig.pvalue`	A number specifies significant cutoff for gene silenced by promoter methylation. Default is 0.01. P value is raw P value without adjustment.

minSubgroupFrac
: A number ranging from 0 to 1 specifying the percentage of samples used to create the groups U (unmethylated) and M (methylated) used to link probes to genes. Default is 0.4 (lowest quintile of all samples will be in the U group and the highest quintile of all samples in the M group).

upstream
: Number of bp upstream of TSS to consider as promoter region

downstream
: Number of bp downstream of TSS to consider as promoter region

save
: A logic. If it is true, the result will be saved.

cores
: Number of cores to be used in paralellization. Default 1 (no paralellization)

Details

promoterMeth

Value

A data frame contains genes whose expression significantly anti-correlated with promoter methylation.

Examples

```
## Not run:
  data(elmer.data.example.promoter)
  Gene.promoter <- promoterMeth(mae.promoter)

## End(Not run)
```

---

render_report *Build report for TCGA.pipe function*

Description

Build HTML report

Usage

```
render_report(
  title = "Report",
  mae.file,
  group.col,
  group1,
  group2,
  direction,
  dir.out,
  genome = "hg38",
  mode = "supervised",
  minSubgroupFrac = 0.2,
  minMetdiff = 0.3,
  metfdr = 0.01,
  permu = 10000,
  rawpval = 0.01,
```

```
  pe = 0.01,
  nprobes = 10,
  lower.OR = 1.1,
  out_file = file.path(getwd(), "report.html"),
  funcivar = FALSE
)
```

Arguments

title	HTML report title
mae.file	Absolute path to the mae used in the analysis (.rda or .rds)
group.col	Group col
group1	Group 1
group2	Group 2
direction	direction used in the analysis
dir.out	Absolute path to folder with results. dir.out used in the analysis
genome	Genome of reference used in the analysis
mode	mode used in the analysis
minSubgroupFrac	minSubgroupFrac used in the analysis
minMetdiff	minMetdiff used in the analysis
metfdr	metfdr used in the analysis
permu	permu used in the analysis
rawpval	rawpval used in the analysis
pe	pe used in the analysis
nprobes	nprobes used in the analysis
lower.OR	lower.OR used in the analysis
out_file	Output file name (i.e report.html)
funcivar	Include funcivar analysis?

Examples

```
## Not run:
render_report(
 group.col = "TN",
 group1 = "Tumor",
 group2 = "Normal",
 dir.out = "~/paper_elmer/Result/BRCA/TN_Tumor_vs_Normal/hypo/",
 direction = "hypo",
 mae.file = "~/paper_elmer/Result/BRCA/BRCA_mae_hg38.rda"
)

## End(Not run)
```

scatter *scatter*

Description

scatter

Usage

```
scatter(
  meth,
  exp,
  legend.title = "Legend",
  category = NULL,
  xlab = NULL,
  ylab = NULL,
  ylim = NULL,
  dots.size = 0.9,
  title = NULL,
  correlation = FALSE,
  correlation.text.size = 3,
  color.value = NULL,
  lm_line = FALSE
)
```

Arguments

meth	A vector of number.
exp	A vector of number or matrix with sample in column and gene in rows.
legend.title	Plot legend title
category	A vector of sample labels.
xlab	A character specify the title of x axis.
ylab	A character specify the title of y axis.
ylim	y-axis limit i.e. c(0,25)
dots.size	Control dots size
title	A character specify the figure title.
correlation	Show spearman correlation values
correlation.text.size	Correlation values
color.value	A vector specify the color of each category, such as
lm_line	A logic. If it is TRUE, regression line will be added to the graph.

Value

A ggplot figure object

scatter.plot *scatter.plot to plot scatter plots between gene expression and DNA methylation.*

Description

scatter.plot is a function to plot various scatter plots between gene expression and DNA methylation. When byPair is specified, scatter plot for individual probe-gene pairs will be generated. When byProbe is specified, scatter plots for one probes with nearby 20 gene pairs will be generated. When byTF is specified, scatter plot for TF expression and average DNA methylation at certain motif sites will be generated.

Usage

```
scatter.plot(data,
             byPair = list(probe = c(), gene = c()),
             byProbe = list(probe = c(), numFlankingGenes = 20),
             byTF = list(TF = c(), probe = c()),
             category = NULL,
             ylim = NULL,
             dots.size = 0.9,
             correlation = FALSE,
             width = 7,
             height = 6,
             dir.out = "./",
             save = TRUE, ...)
```

Arguments

data	A multiAssayExperiment with DNA methylation and Gene Expression data. See createMAE function.
byPair	A list: byPair =list(probe=c(),gene=c()); probe contains a vector of probes' name and gene contains a vector of gene ID. The length of probe should be the same with length of gene. Output see numFlankingGenes
byProbe	A list byProbe =list(probe=c(), geneNum=20); probe contains a vector of probes'name and geneNum specify the number of gene near the probes will ploted. 20 is default for numFlankingGenes Output see detail.
byTF	A list byTF =list(TF=c(), probe=c()); TF contains a vector of TF's symbol and probe contains the a vector of probes' name. Output see detail.
category	A vector labels subtype of samples or a character which is the column name in the colData(data) in the multiAssayExperiment object. Once specified, samples will label different color. The color can be customized by using color.value.
ylim	y-axis limit i.e. c(0,25)
dots.size	Control dots size
correlation	Add pearson correlation values to the plot
width	PDF width
height	PDF height

dir.out	A path specify the directory to which the figures will be saved. Current directory is default.
save	A logic. If true, figure will be saved to dir.out.
...	color.value, lm_line in scatter function

Details

byPair The output will be scatter plot for individual pairs.

byProbe The output will be scatter plot for the probe and nearby genes.

byTF The output will be scatter plot for the TFs and the average DNA methylation at the probes set specified in byTF list.

Value

Scatter plots.

Author(s)

Lijing Yao (maintainer: lijingya@usc.edu)

Examples

```
data <- ELMER:::getdata("elmer.data.example")
scatter.plot(data,
            byProbe=list(probe=c("cg19403323"),numFlankingGenes=20),
            category="definition", save=FALSE)
scatter.plot(data,byProbe=list(probe=c("cg19403323"),numFlankingGenes=20),
            category="definition", save=TRUE) ## save to pdf
# b. generate one probe-gene pair
scatter.plot(data,byPair=list(probe=c("cg19403323"),gene=c("ENSG00000143322")),
             category="definition", save=FALSE,lm_line=TRUE)
```

---

schematic.plot *schematic.plot to plot schematic plots showing the locations of genes and probes.*

Description

schematic.plot is a function to plot schematic plots showing the locations of genes and probes.

Usage

```
schematic.plot(data,
               group.col = NULL,
               group1 = NULL,
               group2 = NULL,
               pair,
               byProbe,
               byGeneID,
               byCoordinate=list(chr=c(), start=c(), end=c()),
               statehub.tracks,
               dir.out="./",
               save=TRUE,...)
```

Arguments

data	A Multi Assay Experiment object with DNA methylation and gene expression Summarized Experiment objects
group.col	A column defining the groups of the sample. You can view the available columns using: colnames(MultiAssayExperiment::colData(data)).
group1	A group from group.col. ELMER will run group1 vs group2. That means, if direction is hyper, get probes hypermethylated in group 1 compared to group 2.
group2	A group from group.col. ELMER will run group1 vs group2. That means, if direction is hyper, get probes hypermethylated in group 1 compared to group 2.#' @param byProbe A vector of probe names.
pair	A data frame with three columns: Probe, Gene ID (Ensemble gene ID) and Pe (empirical p-value). This is the ouput of get.pair function.
byProbe	A vector of probe names
byGeneID	A vector of gene ID
byCoordinate	A list contains chr, start and end. byCoordinate=list(chr=c(),start=c(),end=c()).
statehub.tracks	Relative path to a statehub track.
dir.out	A path specify the directory for outputs. Default is current directory
save	A logic. If true, figures will be saved to dir.out.
...	Parameters for GetNearGenes

Details

byProbes: When a vector of probes' name are provided, function will produce schematic plots for each individual probes. The schematic plot contains probe, nearby 20 (or the number of gene user specified.) genes and the significantly linked gene to the probe.

byGene: When a vector of gene ID are provided, function will produce schematic plots for each individual genes. The schematic plot contains the gene and all the significantly linked probes.

byCoordinate: When a genomic coordinate is provided, function will produce a schematic plot for this coordinate. The schematic plot contains all genes and significantly linked probes in the range and the significant links.

Examples

```
data <- ELMER:::getdata("elmer.data.example")
pair <- data.frame(Probe = c("cg19403323","cg19403323", "cg26403223"),
                   GeneID = c("ENSG00000196878", "ENSG00000009790", "ENSG00000009790" ),
                   Symbol = c("TRAF3IP3","LAMB3","LAMB3"),
                   Raw.p =c(0.001,0.00001,0.001),
                   Pe = c(0.001,0.00001,0.001))
schematic.plot(data,
               group.col = "definition",
               group1 = "Primary solid Tumor",
               group2 = "Solid Tissue Normal",
               pair = pair,
               byProbe = "cg19403323")
schematic.plot(data,
               group.col = "definition",
               group1 = "Primary solid Tumor",
               group2 = "Solid Tissue Normal",
```

```
                pair = pair,
                byGeneID = "ENSG00000009790")

schematic.plot(data,
                group.col = "definition",
                group1 = "Primary solid Tumor",
                group2 = "Solid Tissue Normal",
                pair = pair,
                byCoordinate = list(chr="chr1", start = 209000000, end = 209960000))
## Not run:
   schematic.plot(data,
                   group.col = "definition",
                   group1 = "Primary solid Tumor",
                   group2 = "Solid Tissue Normal",
                   pair = pair,
                   byProbe = "cg19403323",
                   statehub.tracks = "hg38/ENCODE/mcf-7.16mark.segmentation.bed")

## End(Not run)
```

`Stat.diff.meth` *Stat.diff.meth*

Description

Stat.diff.meth

Usage

```
Stat.diff.meth(
  meth,
  groups,
  group1,
  group2,
  test = t.test,
  min.samples = 5,
  percentage = 0.2,
  Top.m = NULL
)
```

Arguments

`meth`	A matrix contain DNA methylation data.
`groups`	A vector of category of samples.
`group1`	Group 1 label in groups vector
`group2`	Group 2 label in groups vector
`test`	A function specify which statistic test will be used.
`min.samples`	Minimun number of samples to use in the analysis. Default 5. If you have 10 samples in one group, percentage is 0.2 this will give 2 samples in the lower quintile, but then 5 will be used.
`percentage`	A number specify the percentage of normal and tumor samples used in the test.
`Top.m`	A logic. If to identify hypomethylated probe Top.m should be FALSE. hypermethylated probe is TRUE.

Value

Statistic test results to identify differentially methylated probes.

`Stat.nonpara` *U test (non parameter test) for permutation. This is one probe vs nearby gene which is good for computing each probes for nearby genes.*

Description

U test (non parameter test) for permutation. This is one probe vs nearby gene which is good for computing each probes for nearby genes.

Usage

```
Stat.nonpara(
  Probe,
  NearGenes,
  Top = NULL,
  correlation = "negative",
  unmethy = NULL,
  methy = NULL,
  Meths = Meths,
  Exps = Exps
)
```

Arguments

`Probe`	A character of name of Probe in array.
`NearGenes`	A list of nearby gene for each probe which is output of GetNearGenes function.
`Top`	A number determines the percentage of top methylated/unmethylated samples. Only used if unmethy and methy are not set.
`correlation`	Type of correlation to evaluate (negative or positive). Negative (default) checks if hypomethylated region has a upregulated target gene. Positive checks if region hypermethylated has a upregulated target gene.
`unmethy`	Index of U (unmethylated) group.
`methy`	Index of M (methylated) group.
`Meths`	A matrix contains methylation for each probe (row) and each sample (column).
`Exps`	A matrix contains Expression for each gene (row) and each sample (column).

Value

U test results

Stat.nonpara.permu *Stat.nonpara.permu*

Description

Stat.nonpara.permu

Usage

```
Stat.nonpara.permu(
  Probe,
  Gene,
  Top = 0.2,
  correlation = "negative",
  unmethy = NULL,
  methy = NULL,
  Meths = Meths,
  Exps = Exps
)
```

Arguments

Probe	A character of name of Probe in array.
Gene	A vector of gene ID.
Top	A number determines the percentage of top methylated/unmethylated samples. Only used if unmethy and methy are not set.
correlation	Type of correlation to evaluate (negative or positive). Negative (default) checks if hypomethylated region has a upregulated target gene. Positive checks if region hypermethylated has a upregulated target gene.
unmethy	Index of U (unmethylated) group.
methy	Index of M (methylated) group.
Meths	A matrix contains methylation for each probe (row) and each sample (column).
Exps	A matrix contains Expression for each gene (row) and each sample (column).

Value

U test results

summarizeTF *Make MR TF binary table*

Description

This function uses ELMER analysis results and summarizes the MR TF identified in each analysis

Usage

```
summarizeTF(files = NULL, path = NULL, classification = "family", top = FALSE)
```

Arguments

files	Output of get.pairs function: dataframe or file path
path	Directory path with the ELMER results. Files with the following pattern will be selected TF.*with.motif.summary.csv.
classification	Consider subfamily or family classifications
top	Get only the top potential (default) or all potentials

TCGA.pipe *ELMER analysis pipeline for TCGA data.*

Description

ELMER analysis pipeline for TCGA data. This pipeline combine every steps of **ELMER** analyses: get.feature.probe, get.diff.meth, get.pair, get.permu, get.enriched.motif and get.TFs. Every steps' results are saved.

Usage

```
TCGA.pipe(
  disease,
  genome = "hg38",
  analysis = "all",
  wd = getwd(),
  cores = 1,
  mode = "unsupervised",
  Data = NULL,
  diff.dir = "hypo",
  genes = NULL,
  mutant_variant_classification = c("Frame_Shift_Del", "Frame_Shift_Ins",
    "Missense_Mutation", "Nonsense_Mutation", "Splice_Site", "In_Frame_Del",
    "In_Frame_Ins", "Translation_Start_Site", "Nonstop_Mutation"),
  group.col = "TN",
  group1 = "Tumor",
  group2 = "Normal",
  ...
)
```

Arguments

disease	TCGA short form disease name such as COAD
genome	Data aligned against which genome of reference. Options: "hg19", "hg38" (default)
analysis	A vector of characters listing the analysis need to be done. Analysis can be "download","distal.probes","diffMeth","pair","motif","TF.search". Default is "all" meaning all the analysis will be processed.
wd	A path shows working dirctory. Default is "./"
cores	A interger which defines number of core to be used in parallel process. Default is 1: don't use parallel process.

mode	This option will automatically set the percentage of samples to be used in the analysis. Options: "supervised" (use 100% of samples) or "unsupervised" (use 20% of samples).
Data	A path shows the folder containing DNA methylation, expression and clinic data
diff.dir	A character can be "hypo" or "hyper", showing dirction DNA methylation changes. If it is "hypo", get.diff.meth function will identify all significantly hypomethylated CpG sites; If "hyper", get.diff.meth function will identify all significantly hypermethylated CpG sites
genes	List of genes for which mutations will be verified. A column in the MAE with the name of the gene will be created with two groups WT (tumor samples without mutation), MUT (tumor samples w/ mutation), NA (not tumor samples)
mutant_variant_classification	List of TCGA variant classification from MAF files to consider a samples mutant. Only used when argument gene is set.
group.col	A column defining the groups of the sample. You can view the available columns using: colnames(MultiAssayExperiment::colData(data)).
group1	A group from group.col. ELMER will run group1 vs group2. That means, if direction is hyper, get probes hypermethylated in group 1 compared to group 2.
group2	A group from group.col. ELMER will run group1 vs group2. That means, if direction is hyper, get probes hypermethylated in group 1 compared to group 2.
...	A list of parameters for functions: GetNearGenes, get.feature.probe, get.diff.meth, get.pair

Value

Different analysis results.

Examples

```
  data <- ELMER:::getdata("elmer.data.example")
  TCGA.pipe(
    disease = "LUSC",
    data = data,
    analysis = c("diffMeth","pair", "motif","TF.search"),
    mode = "supervised",
    group.col = "definition",
    group1 = "Primary solid Tumor",
    group2 = "Solid Tissue Normal",
    diff.dir = c("hypo"),
    dir.out = "pipe",
    sig.dif = 0.0001,
    pvalue = 1.0,
    min.incidence = 0,
    lower.OR = 0.0
  )
## Not run:
  distal.probe <- TCGA.pipe(disease = "LUSC", analysis="distal.enhancer", wd="~/")
 TCGA.pipe(disease = "LUSC",analysis = "all", genome = "hg19", cores = 1, permu.size=300, Pe=0.01)
  projects <- TCGAbiolinks:::getGDCprojects()$project_id
  projects <- gsub("TCGA-","",projects[grepl('^TCGA',projects,perl=TRUE)])
  for(proj in projects) TCGA.pipe(disease = proj,analysis = "download")
  plyr::alply(sort(projects),1,function(proj) {
       tryCatch({
```

```
        print(proj);
        TCGA.pipe(disease = proj,analysis = c("createMAE"))})
      }, .progress = "text")
  plyr::alply(sort(projects),1,function(proj) {
    tryCatch({
      print(proj);
      TCGA.pipe(disease = proj,
                analysis = c("diffMeth","pair", "motif","TF.search"))})
  }, .progress = "text")

  # Evaluation mutation
  TCGA.pipe(disease = "LUSC",analysis = "createMAE",gene = "NFE2L2")
  TCGA.pipe(
     disease = "LUSC",analysis = c("diffMeth","pair", "motif","TF.search"),
     mode = "supervised",
     group.col = "NFE2L2", group1 = "Mutant", group2 = "WT",
     diff.dir = c("hypo"),
     dir.out = "LUSC_NFE2L2_MutvsWT"
   )

## End(Not run)
```

`TF.rank.plot` *TF.rank.plot to plot the scores (-log10(P value)) which assess the correlation between TF expression and average DNA methylation at motif sites.*

Description

TF.rank.plot is a function to plot the scores (-log10(P value)) which assess the correlation between TF expression and average DNA methylation at motif sites. The the motif relevant TF and top3 TFs will be labeled in a different color.

Usage

```
TF.rank.plot(
  motif.pvalue,
  motif,
  title = NULL,
  TF.label = NULL,
  dir.out = "./",
  save = TRUE,
  cores = 1
)
```

Arguments

`motif.pvalue`	A matrix or a path specifying location of "XXX.with.motif.pvalue.rda" which is output of getTF.
`motif`	A vector of characters specify the motif to plot
`title`	Tite title (the motif will still be added to the title)

TF.label	A list shows the label for each motif. If TF.label is not specified, the motif relevant TF and top3 TF will be labeled.
dir.out	A path specify the directory to which the figures will be saved. Current directory is default.
save	A logic. If true (default), figure will be saved to dir.out
cores	A interger which defines the number of cores to be used in parallel process. Default is 1: no parallel process.

Value

A plot shows the score (-log(P value)) of association between TF expression and DNA methylation at sites of a certain motif.

Author(s)

Lijing Yao (maintainer: lijingya@usc.edu)

Examples

```
library(ELMER)
data <- tryCatch(ELMER:::getdata("elmer.data.example"), error = function(e) {
  message(e)
  data(elmer.data.example, envir = environment())
})
enriched.motif <- list("P53_HUMAN.H11MO.0.A"= c("cg00329272", "cg10097755", "cg08928189",
                                "cg17153775", "cg21156590", "cg19749688", "cg12590404",
                                "cg24517858", "cg00329272", "cg09010107", "cg15386853",
                                "cg10097755", "cg09247779", "cg09181054"))
TF <- get.TFs(data,
              enriched.motif,
              group.col = "definition",
              group1 = "Primary solid Tumor",
              group2 = "Solid Tissue Normal",
              TFs = data.frame(
                     external_gene_name=c("TP53","TP63","TP73"),
                     ensembl_gene_id= c("ENSG00000141510",
                                        "ENSG00000073282",
                                        "ENSG00000078900"),
                     stringsAsFactors = FALSE),
             label="hypo")
TF.meth.cor <- get(load("getTF.hypo.TFs.with.motif.pvalue.rda"))
TF.rank.plot(motif.pvalue=TF.meth.cor,
            motif="P53_HUMAN.H11MO.0.A",
            TF.label=createMotifRelevantTfs("subfamily")["P53_HUMAN.H11MO.0.A"],
            save=TRUE)
TF.rank.plot(motif.pvalue=TF.meth.cor,
            motif="P53_HUMAN.H11MO.0.A",
            save=TRUE)
# Same as above
TF.rank.plot(motif.pvalue=TF.meth.cor,
            motif="P53_HUMAN.H11MO.0.A",
            dir.out = "TFplots",
            TF.label=createMotifRelevantTfs("family")["P53_HUMAN.H11MO.0.A"],
            save=TRUE)
```

`TFsurvival.plot` *Creates survival plot of based on the expression of a TF*

Description

This function will create a survival plot for the samples with higher, midium, low expression of a given transcription factor. By defau;t samples with higher expression are the top 30

Usage

```
TFsurvival.plot(data, TF, xlim = NULL, percentage = 0.3, save = TRUE)
```

Arguments

`data`	A multi assay Experiment with clinical data in the phenotypic data matrix containing the following columns: vital_status, days_to_last_follow_up and days_to_death. Default from GDC and TCGAbiolinks
`TF`	A gene symbol
`xlim`	Limit x axis showed in plot
`percentage`	A number ranges from 0 to 1 specifying the percentage of samples in the higher and lower expression groups. Default is 0.3
`save`	Save plot as PDF

Index