Package 'TRESS'

June 23, 2022

Type Package

Title Toolbox for mRNA epigenetics sequencing analysis

Version 1.2.0

Description This package is devoted to analyzing MeRIP-seq data. Current functionalities include 1. detection of transcriptome wide m6A methylation regions 2. detection of transcriptome wide differential m6A methylation regions.

Imports utils, rtracklayer, Matrix, matrixStats, stats, methods, graphics, GenomicRanges, GenomicFeatures, IRanges, Rsamtools, AnnotationDbi

Depends R (>= 4.1.0), parallel, S4Vectors

biocViews Epigenetics, RNASeq, PeakDetection, DifferentialMethylation

License GPL-3 + file LICENSE

Encoding UTF-8

LazyData false

Suggests knitr, rmarkdown,BiocStyle

VignetteBuilder knitr

RoxygenNote 7.1.1

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

git_branch RELEASE_3_15

git_last_commit d75a72a

git_last_commit_date 2022-04-26

Date/Publication 2022-06-23

Author Zhenxing Guo [aut, cre], Hao Wu [ctb]

Maintainer Zhenxing Guo <zhenxing.guo@emory.edu>

R topics documented:

`Basal` *Bin-level and region-level data from basal mouse brain samples*

Description

A data list containing both bin-level and region-level transcriptome locations and read counts across 7 paired input and IP replicates from basal mouse brain samples. It also contains the size factor of each sample for library size normalization.

Usage

```
data(Basal)
```

Format

A list containing two sublists: "Bins" and "Candidates". In list "Bins", there are,

Bins A dataframe of 1000 obs and 5 variables, containing the transcriptome location for 1000 bins of length 50bps

Counts A data matrix of 1000 obs and 14 variables, caontaining bin-level read counts

sf A numerical vector, containing the size factors of 14 samples estimated from the whole transcriptome using bin-level read counts. ...

In list "Candidates", there are,

Regions A dataframe of 500 obs and 5 variables, containing the transcriptome location of 8011 candidates.

Counts A data matrix of 500 obs and 14 variables, caontaining region-level read counts ...

Note, bins and regions may or may not overlap with each other, as both of them are respectively randomly selected from the whole set of bins and candidate regions. However, both data share the same size factor for each sample.

`CallCandidates` *Call candidate m6A regions or candidate differential m6A regions.*

Description

This function first calls m6A bumps from each pair of input and IP sample using bin-level data. Then, bumps from all input and IP pairs are unioned together to obtain a list of candidate regions.

Usage

```
CallCandidates(Counts, bins,
               WhichThreshold ="fdr_lfc", pval.cutoff = 1e-5,
               fdr.cutoff = 0.05, lfc.cutoff = 0.7,
               windlen = 5,lowcount = 30)
```

Arguments

`Counts`	A data matrix containing bin-level (default 50bp) read counts in both IP and input samples, where the sample order is: input1, ip1, input2, ip2, ...
`bins`	A data frame containing the genomic coordinate of each bin of fixed length.
`WhichThreshold`	A character specifying a criterion to select significant bins in bump finding using an ad hoc algorithm. There are five options: "pval" (only use p-values), "fdr" (only use FDR), "lfc" (only use log fold change), "pval_lfc" (use both p-values and log fold changes) and "fdr_lfc" (use FDR and log fold changes). Default is "fdr_lfc".
`pval.cutoff`	A constant indicating the cutoff for p-value. Default is 1e-05.
`fdr.cutoff`	A constant indicating the cutoff for FDR. Default is 0.05.
`lfc.cutoff`	A constant indicating the cutoff for log fold change. Default is 0.7 for fold change of 2.
`windlen`	An integer specifying the length of consecutive bins used in simple moving average smooth of log fold change. Default is 5.
`lowcount`	An integer to filter out candidate regions with lower read counts in input. Default is 30.

Details

The function involves three steps:

- Perform binomial test for each bin based bin-level counts
- Merge significant bins in each input \& IP pair to form bumps usng: `findBumps`
- Combine bumps from all input \& IP pairs to construct a list of candidate regions.

Value

A list containing

`Regions`	A data frame containng genomic coordinate for each candidate region.
`Counts`	A data matrix containing read counts of all samples for each candidate region.

Examples

```
### A toy example, whose results do not have real applications.
data("Basal")
Candidates = CallCandidates(
    Counts = Basal$Bins$Counts,
    bins = Basal$Bins$Bins
    )
```

`CallDMRs.paramEsti` *Model fitting and parameter estimation by TRESS for each candidate DMR.*

Description

TRESS models the read counts in candidate DMR using hierarchical negative binomial distribution, with methylation level of each DMR linked to multi-factors in the design by a linear framework. This function conducts model fitting, parameter estimation, and the variance-covariance matrix computation.

Usage

```
CallDMRs.paramEsti(counts, sf,
                   model, variable,
                   shrkPhi = TRUE,
                   addsuedo = FALSE)
```

Arguments

counts	A dataframe containing read counts in each candidate DMR across all samples.
sf	A numerical vector of size factors for all samples.
variable	A dataframe containing condition information of all samples.
model	A formula to specify which factor in "variable" to be included in model fitting.
shrkPhi	A logical value indicating whether conducting shringkage estimate for dispersion parameter. Default is TRUE.
addsuedo	A logical value indicating whether or not adding a psuedo count of 5 on raw read counts. Default is FALSE.

Value

This function returns a list containing:

Ratio	A dataframe containing the IP/input ratio from all samples.
loglik	A numerical vector containing the log-likelihood of all DMRs.
Coef	A matrix containing estimates of coefficients in the design.
Cov	A list of variance-covariance matrix estimates for all DMRs.

Examples

```
# A toy example
data(DMR_M3vsWT) # data from TRESS
variable = data.frame(predictor = rep(c("WT", "M3"), c(2, 2)))
model = ~1+predictor
DMRfit = CallDMRs.paramEsti(
    counts = DMR_M3vsWT$Counts,
    sf = DMR_M3vsWT$sf,
    variable = variable,
    model = model
    )
```

CallPeaks.multiRep *m6A peak calling with multiple replicates.*

Description

This function identifies and ranks significant m6A peaks, given candidate regions obtained from multiple paired of input \& IP replicates.

Usage

```
CallPeaks.multiRep(Candidates, mu.cutoff,
                   WhichThreshold = "fdr_lfc",
                   pval.cutoff = 1e-5,
                   fdr.cutoff = 0.05,
                   lfc.cutoff = 0.7)
```

Arguments

Candidates	A list containing: genomic coordinates of each candidate region, read counts and log fold change between IP and input in each candidate region. It also contains the size factor of each sample.
mu.cutoff	A constant specifying the background methylation levels. This is estimated automatically based on the first step of peak calling.
WhichThreshold	A character specifying a threshold for significant peaks. There are three options: "pval" (only use p-values), "fdr" (only use FDR), "lfc" (only use log fold change), "pval_lfc" (use both p-values and log fold changes) and "fdr_lfc" (use FDR and log fold changes). Default is "fdr_lfc".
pval.cutoff	A constant indicating the cutoff for p-value. Default is 1e-05.
fdr.cutoff	A constant indicating the cutoff for FDR. Default is 0.05.
lfc.cutoff	A constant indicating the cutoff for log fold change. Default is 0.7 for fold change of 2.

Details

This function first calls CallPeaks.paramEsti to conduct parameter estimation and hypothesis testing for all candidate m6A regions. Then it filters and ranks candidate regions using respective criteria to obtain a list of significant m6A peaks.

Value

The output is a dataframe whose columns are:

chr	Chromosome number of each peak.
start	The start of genomic position of each peak.
end	The end of genomic position of each peak.
strand	The strand of each peak.
summit	The summit of each peak.
lg.fc	Log fold change between normalized IP and normalized input read counts.
mu	Methylation level of each peak if there are more than one replicates.
mu.var	Estimated variance of methylation level for each peak, when there are more than one replicates.
stats	Wald test statistics of each peak, when there are more than one replicate.
shrkPhi	Shrinkage estimation of mehtylation dispersion for each peak, when there are more than one replicates.
shrkTheta	Shrinkage estimation for scale parameter theta in the gamma distribution, when there are more than one replicates.
pvals	P-value calculated based on the Wald-test.
p.adj	Adjusted p-values using Benjamini-Hochberg procedure.
rSocre	A score used to rank each peak. The higher the score, the higher the rank would be.

Note, there are additional columns with name "*.bam". These columns contain the read counts from respective samples.

Examples

```
### A toy example
data("Basal")
CallPeaks.multiRep(
    Candidates = Basal$Candidates,
    mu.cutoff = 0.5
    )
```

`CallPeaks.oneRep` *m6A peak calling with only one replicate.*

Description

This function conducts peak calling for data when there is only one biological replicate of input and IP sample.

Usage

```
CallPeaks.oneRep(Counts, bins, sf = NULL,
                 WhichThreshold = "fdr_lfc",
                 pval.cutoff = 1e-05, fdr.cutoff = 0.05,
                 lfc.cutoff = 0.7, windlen = 5, lowCount = 10)
```

Arguments

`Counts`	A two-column data matrix containing bin-level read counts for both IP and input samples.
`sf`	A numerical vector containg size factors of both IP and input samples. It can be provided by the user, or automatically estimated using "Counts". Default is NULL.
`bins`	A dataframe containing the genomic locations (chr, start, end, strand) of each bin.
`WhichThreshold`	A character specifying a criterion to select significant bins in bump finding using an ad hoc algorithm. There are five options: "pval" (only use p-values), "fdr" (only use FDR), "lfc" (only use log fold change), "pval_lfc" (use both p-values and log fold changes) and "fdr_lfc" (use FDR and log fold changes). Default is "fdr_lfc".
`pval.cutoff`	A constant indicating a cutoff for p-value. Default is 1e-05.
`fdr.cutoff`	A constant indicating a cutoff for FDR. Default is 0.05.
`lfc.cutoff`	A constant indicating a cutoff for log fold change. Default is 0.7 for fold change of 2.
`windlen`	An integer specifying the length of consecutive bins used in simple moving average smooth of log fold change. Default is 5.
`lowCount`	An integer to filter out m6A regions with lower read counts. Default is 10.

Details

When there is only one replicate, TRESS assigns a p-value for each bin based on the binomial test. Then it calls candidates with the same algorithm used when there are multiple biological replicates. Binomal tests are performed one more time to select significant candidates as final list of peaks.

Value

It returns an excel containing the information for each peak:

`chr`	Chromosome number of each peak.
`start`	The start of genomic position of each peak.
`end`	The end of genomic position of each peak.
`strand`	The strand of each peak.
`summit`	The summit of each peak.
`pvals`	P-value for each peak calculated based on binomial test.
`p.adj`	Adjusted p-values using Benjamini-Hochberg procedure.
`lg.fc`	Log fold change between normalized IP and normalized input read counts.

Note, there are additional columns with name "*.bam". These columns contain the read counts from IP and input samples.

Examples

```
## A toy example
data("Basal")
bincounts = Basal$Bins$Counts[, 1:2]
sf0 = Basal$Bins$sf[1:2]
bins = Basal$Bins$Bins
peaks = CallPeaks.oneRep(Counts = bincounts,
                         sf = sf0, bins = bins)
head(peaks, 3)
```

`CallPeaks.paramEsti` *Parameter estimation in m6A peak calling with multiple replicates.*

Description

This function estimates all involved parameters in Bayesian hierarchical negative binomial model, which is built for read counts from candidate regions generated from multiple input\& IP replicates.

Usage

```
CallPeaks.paramEsti(mat, sf = NULL, cutoff = NULL,
                    update = "Joint",
                    trans = NULL,
                    optM = "L-BFGS-B",
                    myfscale = -1e+06)
```

Arguments

mat	A matrix containing read counts from all paired input \& input replicates. The order of samples are: input1, IP1, input2, IP2,...
sf	A vector of size factors for each sample. It can be provided by the users or estimated automatically from the data. Default is NULL.
cutoff	Background methylation level, which can be automatically estimated based on the background read counts in IP and input samples, or provided by users. Defauls is NULL.
update	A logical value indicating whether jointly estimating the nuisance parameter theta with dispersion parameter phi listed in the proposed model. Possible options are "OnlyPhi", "Iterative" and "Joint". "OnlyPhi" means only updating phi_i using R function optimize while fixing parameter theta as the plug-in moment estimator; "Iterative" means iteratively updating and phi using R function optimize; "Joint" means updating them together using R function optim. Default is "Joint".
optM	A charactor value to specify which optimization algorithm used in the R function optim. The options are: "Nelder-Mead", "BFGS", "CG", "L-BFGS-B", "SANN" and "Brent". Default is "L-BFGS-B". See more details in help pages of optim.
trans	Needed when **optM == "Nelder-Mead"**. It specifies which transformation function used in the estimation of dispersion and/or theta parameter(s) which are subjected to the nonnegative constraints. Possible options are "sin()" and "exp()". Default is NULL.
myfscale	A stop criteria in optim. Default is -1e+06.

Details

This function mainly involves three estimation procedures:

- Estimate methylation levels
- Estimate dispersion parameters and the variance of the estimated methylation levels
- Calculate test statistics and p-values. Also, it calculates a score used for peak ranking.

Value

mu	Estimation of methylation levels of all peaks.
mu.var	Estimated variance for estimated methylation level.
shrkPhi	Shrinkage estimator for dispersion parameter phi_i.
shrkTheta	Shrinkage estimator for parameter theta_i if update == "Joint" or "Iterative". Otherwise it would be a plug-in moment estimator.
stats	Wald-test statisitcs.
pvals	P-values derived from normal distribution based on the Wald-test statisitcs.
p.adj	Adjusted p-values using Benjamini-Hochberg procedure.
rSocre	A score used to ranke each region. The higher the score, the higher the rank would be.

Examples

```
### A toy example using basal samples from mouse cortex
data("Basal")
res = CallPeaks.paramEsti(
    mat = as.matrix(Basal$Candidates$Counts),
    sf = Basal$Bins$sf,
    cutoff = 0.5
    )
```

CoefName *Obtain the name of each coefficient in design matrix.*

Description

This functions returns the name of each coefficient in the design, for the convenience of constructing correct contrast for hypothesis testing.

Usage

```
CoefName(DMR)
```

Arguments

DMR	A list which is the output from TRESS_DMRfit.

Value

A character vector containing the name of each coefficient in design matrix.

Examples

```
data("DMR_SixWeekvsTwoWeek")
design = data.frame(time = rep(c("2wk", "6wk"), each = 4),
                    region = rep(rep(c("Cortex", "Hypothalamus"),
                                     each = 2),2)
                    )
model = ~1 + time + region + time*region
DMRfit = CallDMRs.paramEsti(
    counts = DMR_SixWeekvsTwoWeek$Counts,
    sf = DMR_SixWeekvsTwoWeek$sf,
    variable = design,
    model = model
    )
CoefName(DMRfit)
```

DivideBins *Obtain genomic bins and bin-level read counts from BAM files.*

Description

This function first divides the whole genome into equal-sized bins and then calculates read counts in each bin for all samples. The number of bins depends on the input annotation file, bin size and whether or not including intronic regions.

Usage

```
DivideBins(IP.file, Input.file, Path_To_AnnoSqlite,
           InputDir,OutputDir, experimentName,
           binsize = 50, filetype = "bam",
           IncludeIntron = FALSE)
```

Arguments

IP.file	A vector of characters containing the name of BAM files for all IP samples.
Input.file	A vector of characters containing the name of BAM files for all input control samples.
Path_To_AnnoSqlite	A character to specify the path to a "*.Sqlite" file used for genome annotation.
experimentName	A character to specify a name for output results.
binsize	A numerical value to specify a size of window to bin the genome and get bin-level read counts. Default value is 50.
filetype	A character to specify the format of input data. Possible choices are: "bam", "bed" and "GRanges". Default is "bam".
InputDir	A character to specify the input directory of all BAM files.
OutputDir	A character to specify an output directory to save all results. Default is NA, which will not save any results.
IncludeIntron	A logical value indicating whether to include (TRUE) intronic regions or not (False). Default is FALSE.

Value

The value returned by this function is a list containing two components:

bins	A dataframe containing the genomic cordinates of all bins.
binCount	A M-by-N matrix containg bin-level read counts in M bins and N samples, where N is two times of the length of "IP.file" or "Input.file". The column order depends on the sample order in "Input.file" and "IP.file".

If the "OutputDir" is specified, then both genomic bins and corresponding bin-level read counts would be saved as an ".rda" file.

Examples

```
# use data in pakage datasetTRES
# available on github, which can be installed by
# install_github("https://github.com/ZhenxingGuo0015/datasetTRES")
## Not run:
library(datasetTRES)
IP.file = c("cb_ip_rep1_chr19.bam", "cb_ip_rep2_chr19.bam")
Input.file = c("cb_input_rep1_chr19.bam", "cb_input_rep2_chr19.bam")
BamDir = file.path(system.file(package = "datasetTRES"), "extdata/")
Path_sqlit = file.path(system.file(package = "datasetTRES"),
 "extdata/mm9_chr19_knownGene.sqlite")
#OutDir = "/Users/zhenxingguo/Documents/research/m6a/packagetest"
allBins = DivideBins(
    IP.file = IP.file,
    Input.file = Input.file,
    Path_To_AnnoSqlite = Path_sqlit,
    InputDir = BamDir
    )

## End(Not run)
```

DMRInfer *P-value calculation given Wald statistics.*

Description

This function calculates p-values for candidate DMRs given Wald statistics, using respectively two-component mixtures of normal, truncated normal and standard normal distributions.

Usage

```
DMRInfer(stat, nullModel = "standN")
```

Arguments

stat	A vector containing Wald statistics of all candidate DMRs.
nullModel	A character to specify a method to calculate p-value based on the statistics. It can be "standN", "2mix" and "trunN" for standard normal, two-component mixed gaussian and truncated normal respectively. Defult is "standN".

Details

In addition to standard normal distribution, TRESS provides another two distributions to calculate p-values given Wald statistics, in case that statistics are inflated by potentially underestimated dispersion in data.

One is two-component mixtures of normal distribution, where TRESS assumes that,

$$T_w \sim pN(0, \sigma_0^2) + (1-p)N(0, \sigma_1^2)$$

where T_w is Wald statistics, σ_0 and σ_1 are standard deviations of distribution that T_w follows under null and alternative hypothesis in TRESS_DMRtest. P-values for T_w are calculated using $N(0, \hat{\sigma}_0^2)$.

The other one is truncated normal distribution, where TRESS assumes that,

$$tT_w \sim pN(0, \sigma^2)$$

where $tT_w = T_w \in [-b, b]$. Here, T_w within range [-b, b] is assumed to sampled from null distribuion $N(0, \sigma^2)$. For this truncated normal distribution, TRESS explores different values for boundary b ranging from 1.5 to 2 by step 0.1. TRESS estimates a $\hat{\sigma}$ for each of 6 boundaries. If $\hat{\sigma}_{max} - \hat{\sigma}_{min} > 0.5$, TRESS calculates p-values for T_w using $N(0, \hat{\sigma}_{min}^2)$. Otherwise, p-values are obtained using $N(0, \hat{\sigma}^2)$, with $\hat{\sigma}$ estimated under $b = 2$.

Value

This function returns a dataframe containing p-values and Benjamini-Hochberg procedure adjusted p-values.

Examples

```
### use a randomly generated toy data as an
### illustrate of DMRInfer
set.seed(12345)
p = 0.8
nsites = 10000
flag.TP = rep(NA, nsites)
T_w = rep(NA, nsites)
for (i in seq_len(nsites)) {
  u = runif(1, min = 0, max = 1)
  if(u < p){
    flag.TP[i] = FALSE
    T_w[i] = rnorm(1, 0, sd = 1)
  }else{
    flag.TP[i] = TRUE
    T_w[i] = rnorm(1, 0, sd = 5)
  }
}
res = DMRInfer(stat = T_w, nullModel = "standN")
sum(res$padj < 0.05 & !flag.TP)/sum(res$padj < 0.05)

res = DMRInfer(stat = T_w, nullModel = "2mix")
sum(res$padj < 0.05 & !flag.TP)/sum(res$padj < 0.05)

res = DMRInfer(stat = T_w, nullModel = "trunN")
sum(res$padj < 0.05 & !flag.TP)/sum(res$padj < 0.05)
```

DMR_M3vsWT *Transcriptome location and read counts of 200 candidate DMRs, and size factors.*

Description

A dataset containing the transcriptome location, read counts of 200 candidate DMRs from 4 samples. Each sample contains two paired of IP and input replicates. It also contains size factors estimated from the whole transcriptome to normalize sequencing depth of each sample. In addition, a small proportion of transcriptome bins and their read counts are also included for the purpose of visualizing individual DMR.

Usage

```
data(DMR_M3vsWT)
```

Format

A list with 5 elements

Regions A dataframe of 200 obs and 5 variables, containing the transcriptome location of each candidate DMR.

Counts A dataframe of 200 obs and 8 variables, caontaining the read counts for candidate DMRs. The counts are from both Wild type and METTL3-Knockout samples. Each has two paired of IP and input replicates.

sf A numerical vector, containing the size factors of each sample estimated from the whole transcriptome

Bins A dataframe of 737 obs and 5 variables, containing the transcriptome location of bins overlapping with a small subset of 200 candidate DMRs.

BinsCounts Read counts (200-by-8) correspond to "Bins". ...

DMR_SixWeekvsTwoWeek *Transcriptome location and read counts of 200 candidate DMRs.*

Description

A data list which consists of the transcriptome location and read counts of 200 candidate DMRs from 8 mouse brain samples. Each sample has two paired IP and input replicates. It also contains the size factor of each sample and the estimated methylation ratio in each sample for 200 canidate DMRs.

Usage

```
data(DMR_SixWeekvsTwoWeek)
```

Format

A list with 4 elements

Regions A dataframe of 200 obs and 5 variables, containing the transcriptome location for 200 candidate DMRs from two mouse brain regions at two time points.

Counts A dataframe of 200 obs and 16 variables, caontaining candidate DMR read counts in each paired IP and input replicate of all 8 samples.

sf A numerical vector, containing the size factors of all samples estimated from the whole transcriptome

MeRatio A data matrix of 200-by-8, containing the estimated methylation ratio for each sample. ...

`filterRegions` *Pre-filtering of candidate DMRs.*

Description

This function filters out candidate DMRs who have small (e.g., less than 25% quantile) marginal coefficient of variation (CV) in methylation ratio.

Usage

```
filterRegions(Candidates, quant = 0.25)
```

Arguments

`Candidates`	A list containing "Counts", "Regions", "sf" for read counts, genomic coordinate of each candidate DMR and size factor of all samples. See output of `CallCandidates`.
`quant`	A percentage to specify a quantile cutoff. Regions with CV smaller than this quantile would be filtered out. Default is 25%.

Value

A list with the same structure with input "Candidates" but with a smaller number of candidate DMRs.

Examples

```
# A toy example
data(DMR_M3vsWT) # data from TRESS
sub.DMR_M3vsWT = filterRegions(DMR_M3vsWT)
```

findBumps *Bump-finding from transcriptome bins.*

Description

This function constructs transcriptome m6A bumps for each input \& IP replicate, by merging together bins having significant enrichment of IP over input control reads.

Usage

```
findBumps(chr, pos, strand, x, count,
          use = "pval",
          pval.cutoff,
          fdr.cutoff,
          lfc.cutoff,
          sep = 2000,
          minlen = 100,
          minCount = 3,
          dis.merge = 100,
          scorefun = mean,
          sort = TRUE)
```

Arguments

chr	Chromosome number of all bins.
pos	Transcriptome start position of all bins.
strand	Strand of all bins.
x	A dataframe containing the p-values, fdrs and log fold changes of all bins.
count	Read counts in each bin from paired input and IP sample.
use	A character to specify which criterion to select significant bins. It takes among "pval", "fdr", "lfc", "pval_lfc" and "fdr_lfc". "pval": The selection is only based on P-values; "fdr": The selection is only based on FDR; "lfc": The selection is only based on log fold changes between normalized IP and normalized input read counts; "pval_lfc": The selection is based on both p-values and log fold changes; "fdr_lfc": The selection is based on both FDR and log fold changes. Default is "pval".
pval.cutoff	A numerical value to specify a cutoff for p-value. Default is 1e-5.
fdr.cutoff	A numerical value to specify a cutoff for fdr. Default is 0.05.
lfc.cutoff	A numerical value to specify a cutoff for log fold change between normalized IP and input read counts. Default is 0.7 for fold change of 2.
sep	A constant used divide genome into consecutive sequenced regions. Any two bins with distance greater than sep will be grouped into different regions. Default is 2000.

minlen	A constant to select bumps who have minimum length of minlen. Default is 100.
minCount	A constant to select bumps who have at least minlen number of bins. Default is 3.
dis.merge	A constant. Any twp bumps with distance smaller than dis.merge would be merged. Default is 100.
scorefun	A character indicating a function used to assign a score for each bump base on p-values of all spanned bins. Default is "mean", meaning that the score is an average of bin-level p-values.
sort	A logical value indicating whether rank (TRUE) bumps with the score output from scorefun or not (FALSE). Default is TRUE.

Value

This function returns a dataframe containing the chromosome, start position, end position, length, strand, summit, total read counts (both IP and input) and score of each bump.

Examples

```
### Use example dataset "Basal" in TRESS
### to illustrate usage of this function
data("Basal")
bins = Basal$Bins$Bins
Counts = Basal$Bins$Counts
sf = Basal$Bins$sf
colnames(Counts)
dat = Counts[, 1:2]
thissf = sf[1:2]
### pvals based on binomial test
idx = rowSums(dat) > 0
Pvals = rep(1, nrow(dat))
Pvals[idx] = 1 - pbinom(dat[idx, 2],
                        rowSums(dat[idx, ]),
                        prob = 0.5)
### lfc
c0 = mean(as.matrix(dat), na.rm = TRUE)  ### pseudocount
lfc = log((dat[, 2]/thissf[2] + c0)/(dat[, 1]/thissf[1] + c0))
x.vals = data.frame(pvals = Pvals,
                    fdr = p.adjust(Pvals, method = "fdr"),
                    lfc = lfc)
### find bumps based on pvals, fdr or lfc
Bumps = findBumps(chr = bins$chr,
                  pos = bins$start,
                  strand = bins$strand,
                  x = x.vals,
                  use = "fdr_lfc",
                  fdr.cutoff = 0.01,
                  lfc.cutoff = 0.5,
                  count = dat)
head(Bumps, 3)
```

meRatio *Observed m6A methylation ratio.*

Description

This function calculates, for each candidate region, the enrichment of normalized IP read counts versus the sum of normalized IP and input control read counts.

Usage

```
meRatio(counts, sf)
```

Arguments

counts	A data matrix containing read counts in each region across sample input1, ip1, input2, ip2, input3, ip3, ...
sf	A numerical vector containing the size factor of each sample, which is used for sequencing depth normalization. The sample order here is the same as that in counts.

Value

ratio	A numerical data matrix containing the methylation ratio of each candidate region across all samples. Here, the number of columns is half of the number of columns in read count matrix.

Examples

```
data("Basal")
## methylatinon ratio
Ratio = meRatio(
    counts = Basal$Candidates$Counts,
    sf = Basal$Bins$sf
    )
head(Ratio, 3)
```

ShowOnePeak *Visulization of a single peak along the genome.*

Description

This function plots the estimated methylation level (as bars) of each bin within a peak for each replicate, and the corresponding normalized input read depth (grey curve).

Usage

```
ShowOnePeak(onePeak, allBins, binCounts,
            isDMR = FALSE,Sname = NULL,
            ext = 500, ylim = c(0, 1))
```

Arguments

onePeak	A one-row dataframe containing the genomic position of a single peak: chr, start, end, strand.
allBins	A dataframe contaning genomic position of all bins used to call peaks: chr, start, end, strand.
binCounts	A dataframe contaning the read counts of all bins for each replicate. The sample order is: input1, ip1, input2, ip2, ...
isDMR	A logical value indicating whether the input region is DMR. Default is FALSE.
Sname	Sample names. If isDMR = TRUE, then it will be used as the title of each plot.
ext	An integer indicating the length of base paris to extend the region on both sides: (start - ext, end + ext). Default is 500.
ylim	The range of y-axis to plot. Default is c(0, 1)

Value

It only generates a plot. No specific output.

See Also

ShowOneDMR from "DSS" package.

Examples

```
### read peaks
peaks = read.table(file.path(system.file(package = "TRESS"),
                             "extdata/examplebyBam_peaks.xls"),
                   sep = "\t", header = TRUE)
### load annotation and bin counts
load(file.path(system.file(package = "TRESS"),
               "extdata/examplebyBam.rda"))
 allBins = as.data.frame(bins$bins)
 colnames(allBins)[1] = "chr"
 allBins$strand = binStrand
 for (i in 1:4) {
   ShowOnePeak(
   onePeak = peaks[i,],
   allBins = allBins, binCounts = allCounts
   )
 }
```

TRESS_DMRfit *Differential m6A methylation analysis for MeRIP-seq data under general experimental design*

Description

This function performs differential m6A analysis through the following three steps:

- Divide the whole genome to obtain bin-level read counts: DivideBins
- Call candidate differential m6A methylation regions (DMRs): CallCandidates
- Model fitting on candidate DMRs based on Negative Binomial distribution: CallDMRs.paramEsti

Usage

```
TRESS_DMRfit(IP.file, Input.file, Path_To_AnnoSqlite,
             variable = NULL, model = NULL,
             InputDir, OutputDir = NA,
             experimentName = NA,
             binsize = 50,
             filetype = "bam",
             IncludeIntron = FALSE,
             filterRegion = TRUE,
             shrkPhi = TRUE,
             addsuedo = FALSE)
```

Arguments

IP.file	A vector of characters containing the name of BAM files for all IP samples.
Input.file	A vector of characters containing the name of BAM files for all input control samples.
Path_To_AnnoSqlite	A character to specify the path to a "*.sqlite" file used for genome annotation.
variable	A dataframe containing condition information of all samples. Default is NULL.
model	A formula to specify which factor in "variable" will be included into design for model fitting. Default is NULL.
InputDir	A character to specify the input directory of all BA, files.
OutputDir	A character to specify an output directory to save bin-level and region-level data. Default is NA, which will not save any results.
experimentName	A character to specify the name of results if "OutputDir" is provided.
binsize	A numerical value to specify the size of window to bin the genome. Default value is 50.
filetype	A character to specify the format of input data. Possible choices are: "bam", "bed" and "GRanges". Default is "bam".

IncludeIntron	A logical value indicating whether to include (TRUE) intronic regions or not (False). Default is FALSE.
filterRegion	A logical value indicating whether to filter out candidate DMRs based on their marginal coefficient of variation (CV) in methylation ratios. If TRUE, then a candidate DMR with CV < 25% quantile would be filtered out. Default value is TRUE.
shrkPhi	A logical value to indicate whether conducting shringkage estimate for dispersion parameter. Default is TRUE.
addsuedo	A logical value to indicate whether or not adding a psuedo count 5 on raw read counts. Default is FALSE.

Details

For complete details on each step (especially step 3) in above "Description" section, please see the manual pages of respective functions.

Value

This function generates three sets of results: "allBins", "Candidates" and "DMRfit" returned respectively by function DivideBins, CallCandidates and CallDMRs.paramEsti. If "OutputDir" is not specified, only "DMRfit" will be returned. If "OutputDir" is specified, in addition to returning "DMRfit", "allBins" and "Candidates" will also be saved under the provided output directory.

Detailed structure of "DMRfit", "Candidates" and "allBins" can be found in the manual of respective functions.

Author(s)

Zhenxing Guo <zhenxing.guo@emory.edu>

References

Zhenxing Guo, Andrew M. Shafik, Peng Jin, Hao Wu. "Differential RNA Methylation Analysis for MeRIP-seq Data under General Experimental Design."

See Also

DivideBins, CallCandidates, CallDMRs.paramEsti

Examples

```
## Not run:
Input.file = c("input1.bam", "input2.bam",..., "inputN.bam")
IP.file = c("ip1.bam", "ip2.bam", ..., "ipN.bam")
InputDir = "/directory/to/BAMfile"
OutputDir = "/directory/to/output"
Path_sqlit = "/path/to/xxx.sqlite"
design = "YourDesign"
model = "YourModel"
DMR.fit = TRESS_DMRfit(IP.file = IP.file,
```

```
                        Input.file = Input.file,
                        Path_To_AnnoSqlite = Path_sqlit,
                        variable = design,
                        model = model,
                        InputDir = InputDir,
                        OutputDir = OutputDir,
                        experimentName = "example"
                        )

## End(Not run)
```

TRESS_DMRtest *Hypothesis testing on candidate DMRs.*

Description

This function conducts statistical test for each candidate DMR based on user specified contrast of coefficients in design.

Usage

```
TRESS_DMRtest(DMR, contrast, nullModel = "standN")
```

Arguments

DMR	A list at least containing IP/input ratio, the coefficients estimate, variance-covariance estimate. This can be obtained from the ouput of TRESS_DMRfit.
contrast	A contrast for all coefficients in the design. It can be either a (p+1) vector or a m-by-(p+1) matrix, where p is the number of columns in the design. m depends on the number of relationships that users want to test.
nullModel	A character to specify a method to calculate p-value based on the statistics. It can be "standN", "2mix" and "trunN" for standard normal, two-component mixed gaussian and truncated normal respectively. Defult is "standN".

Details

The hypothesis for each of candidate DMR i is of the form:

$$H_0 : C^T R_i = 0 vs. H_1 : C^T R_i \neq 0$$

where C is a contrast of all coefficients in model design; R_i is coefficient vector for DMR i. If the C is a vector, then TRESS performs Wald test; if the C is a matrix, then TRESS conducts F-test.

Value

This function returns a dataframe containing the testing results for specified contrast. The columns are

baseMean	Averaged methylation level cross all samples.
logOR	Estimated value of contrast: $C^T R_i$. Only available if constrast is a vector.
lorSE	Standard error of $C^T R_i$. Only available if constrast is a vector.
stat	Test statistics.
pvalue	P-values from statistical tests.
padj	Benjamini-Hochberg procedure adjusted p-values.

Author(s)

Zhenxing Guo <zhenxing.guo@emory.edu>

References

Zhenxing Guo, Andrew M. Shafik, Peng Jin, Hao Wu. "Differential RNA Methylation Analysis for MeRIP-seq Data under General Experimental Design."

Examples

```
# A toy example
data(DMR_M3vsWT) # data from TRESS
variable = data.frame(predictor = rep(c("WT", "M3"), c(2, 2)))
model = ~1+predictor
DMR.fit = CallDMRs.paramEsti(
    counts = DMR_M3vsWT$Counts,
    sf = DMR_M3vsWT$sf,
    variable = variable,
    model = model
    )
DMR.test = TRESS_DMRtest(DMR = DMR.fit, contrast = c(0, 1))
head(DMR.test, 3)
head(DMR_M3vsWT$Regions[which(DMR.test$padj < 0.05), ], 3)
```

TRESS_peak *Detecting m6A methylation regions from Methylated RNA Immunoprecipitation Sequencing.*

Description

This is a wrapper function to call m6A peaks transcriptome wide. When there are multiple biological replicates, it

- Divides the whole genome to obtain bin-level read counts: DivideBins
- Calls candidate m6A methylation regions: CallCandidates
- Model fitting on candidate peaks based on Negative Binomial distribution: CallPeaks.multiRep

If there is only one replicate, it calls CallPeaks.oneRep to detect m6A methylation regions.

Usage

```
TRESS_peak(IP.file, Input.file, Path_To_AnnoSqlite,
           binsize = 50,
           WhichThreshold = "fdr_lfc",
           pval.cutoff0 = 1e-5,
           fdr.cutoff0 = 0.05,
           lfc.cutoff0 = 0.7,
           lowcount = 30,
           InputDir,
           OutputDir = NA,
           experiment_name,
           filetype = "bam",
           IncludeIntron = FALSE)
```

Arguments

IP.file	A vector of characters containing the name of BAM files for all IP samples.
Input.file	A vector of characters containing the name of BAM files for all input control samples.
Path_To_AnnoSqlite	A character to specify the path to a "*.sqlite" file used for genome annotation.
binsize	A numerical value to specify the size of window to bin the genome and get bin-level read counts. Default value is 50.
WhichThreshold	A character to specify which criterion to select significant bins in the first step, and also significant m6A regions in the second step. It takes among "pval", "fdr", "lfc", "pval_lfc" and "fdr_lfc". "pval": The inference is only based on P-values; "fdr": The inference is only based on FDR; "lfc": The inference is only based on log fold changes between normalized IP and normalized input read counts; "pval_lfc": The inference is based on both p-values and log fold changes; "fdr_lfc": The inference is based on both FDR and log fold changes. Default is "fdr_lfc".
pval.cutoff0	A numerical value to specify a cutoff for p-value. Default is 1e-5.
fdr.cutoff0	A numerical value to specify a cutoff for fdr. Default is 0.05.
lfc.cutoff0	A numerical value to specify a cutoff for log fold change between normalized IP and input read counts. Default is 0.7 for fold change of 2.
lowcount	An integer to filter out regions with total input counts < lowcount. Default is 30.
InputDir	A character to specify the input directory of all BAM files.
OutputDir	A character to specify an output directory save all results. Default is NA, which will not save any results.
experiment_name	A character to specify the name of results.
filetype	A character to specify the format of input data. Possible choices are: "bam", "bed" and "GRanges". Default is "bam".
IncludeIntron	A logical value indicating whether to include (TRUE) intronic regions or not (False). Default is FALSE.

Details

TRESS implements a two-step procedure to conduct peak calling for MeRIP-seq data with multiple biological replicates. In the first step, it quickly divide the whole genome into equal sized bins and loosely indentifies candidate peak regions using an ad hoc procedure. In the second step, it detects high confident peaks among candidate regions and ranks them with more rigorous statistical modeling based on an empirical Bayesian hierarchical model.

When there is only one biological replciate, candidate regions from the above two-step procedure will be output as the final list of peaks. P-values come from binomial test, which are further adjusted using Benjamini-Hochberg procedure.

Value

If directory OutputDir is specified, this function will output two sets of results. One is saved as ".rda", which contains all bin-level data (genome coordinates and read counts matrix). The other one is an ".xls" file, which contains information of all peaks. The columns of the peak excel files are:

`chr`	Chromosome number of each peak.
`start`	The start of genomic position of each peak.
`end`	The end of genomic position of each peak.
`strand`	The strand of each peak.
`summit`	The summit of each peak.
`lg.fc`	Log fold change between normalized IP and normalized input read counts for each peak.
`pvals`	P-values calculated based on the Wald-test.
`p.adj`	Adjusted p-values using Benjamini-Hochberg procedure.

If there are multiple replicates, the excel will also include following columns:

`mu`	Estimated methylation level of each peak.
`mu.var`	Estimated variance for methylation level of each peak
`stats`	Wald test statistics of each peak
`shrkPhi`	The shrinkage estimation of dispersion for mehtylation levels of each peak.
`shrkTheta`	The shrinkage estimation for scale parameter theta in the gamma distribution.
`rSocre`	A score defined by TRESS to rank each peak. The higher the score, the higher the rank would be.

Note, there are additional columns regardless of the number of replicates. Those columns contain read counts from respective samples and have names "*.bam".

Author(s)

Zhenxing Guo <zhenxing.guo@emory.edu>

References

Guo, Z., Shafik, A. M., Jin, P., Wu, Z., and Wu, H. (2021) Detecting m6A methylation regions from Methylated RNA Immunoprecipitation Sequencing. Bioinformatics, 1-7. https://doi-org.proxy.library.emory.edu/10.1093/bioinformatics/btab181

Examples

```
## Use BAM files in datasetTRES
# install_github("https://github.com/ZhenxingGuo0015/datasetTRES")
## Not run:
library(datasetTRES)
IP.file = c("cb_ip_rep1_chr19.bam", "cb_ip_rep2_chr19.bam")
Input.file = c("cb_input_rep1_chr19.bam", "cb_input_rep2_chr19.bam")
BamDir = file.path(system.file(package = "datasetTRES"), "extdata/")
annoDir = file.path(
  system.file(package = "datasetTRES"),
  "extdata/mm9_chr19_knownGene.sqlite"
  )
OutDir = "/directory/to/output"
 TRESS_peak(IP.file = IP.file,
           Input.file = Input.file,
           Path_To_AnnoSqlite = annoDir,
           InputDir = BamDir,
           OutputDir = OutDir,
           experiment_name = "examplebyBam",
           filetype = "bam")
peaks = read.table(paste0(OutDir, "/", "examplebyBam_peaks.xls"),
                   sep = "\t", header = TRUE)

## End(Not run)
```

Index