Package ‘methyltools’

November 26, 2015

Type Package

Title Additional tools for the analysis of methylation data

Version 0.6

Date 2014-06-23

Author Pierre Bady <pierre.bady@unil.ch>

Maintainer Pierre Bady <pierre.bady@unil.ch>

Depends R (>= 3.2.0), CGHcall, CGH-
base, minfi,preprocessCore, org.Hs.eg.db, TxDb.Hsapiens.UCSC.hg19.knownGene, lumi,mixtools, snow-
fall,RPMM,ade4

Suggests boot

Description Additional tools for the analysis of methylation data

License GPL (>= 2)

R topics documented:

`methyltools-package` *Additional tools for the analysis of methylation data*

Description

Additional tools for the analysis of methylation data

Details

Package:	methyltools
Type:	Package
Version:	0.1
Date:	2013-03-15
License:	GPL (>= 2)

Author(s)

Author: Pierre Bady <pierre.bady@unil.ch>
Maintainer: Pierre Bady <pierre.bady@unil.ch>

References

+ add reference related to lumi and minfi
Staaf, Johan and Vallon-Christersson, Johan and Lindgren, David and Juliusson, Gunnar and Rosenquist, Richard and Hoglund, Mattias and Borg, Ake and Ringner, Markus (2008) Normalization of Illumina Infinium whole-genome SNP data improves copy number estimates and allelic intensity ratios,BMC Bioinformatics, 9:409
Louhimo, R. and Hautaniemi, S. (2011) CNAmet: an R package for integration of copy number, expression and methylation data Bioinformatics 27(6):887-888.

`auc` *Area Under Curves*

Description

This function compute the area under curves.

Usage

```
auc(x,y)
```

Arguments

`x`	a numeric vector
`y`	a numeric vector

Value

The function returns a numeric value corresponding to the area under the curves.

Examples

```
# y <- rpois(1:10)
# d1 <- density(y)
# auc(d1$x,d1$y)
```

`beta3mix` *set of functions related to beta3mix object*

Description

set of functions associated with the beta3mix objects, including print, simulate and graphical representation.

Usage

```
beta3mix(x,n=10000,initial=c(0.2,0.75),niter=25,tol=1e-4,...)
beta3mix.light(x,n=10000,initial=c(0.2,0.75),niter=25,tol=1e-4,...)
## S3 method for class 'beta3mix'
print(x,...)
## S3 method for class 'beta3mix'
plot(x,data,nsim=10000,col=c("black","green2"),...)
## S3 method for class 'beta3mix'
predict(object,data,option="postcluster",encoding=NULL,...)
## S3 method for class 'beta3mix'
simulate(object,nsim=1,seed=NULL,...)
```

Arguments

`x`	a numeric vector or object 'beta3mix'
`object`	an object 'beta3mix'
`data`	a numeric vector
`n`	a numeric value corresponding to the number of values used to fit the model (10000 by default)
`nsim`	a numeric value corresponding to the number of simulated values
`initial`	initial condition used to defined a priori classification
`niter`	a numeric value corresponding to the maximum number of iteration
`tol`	a numeric value corresponding to the tolerance criteria
`cpus`	a numeric value providing the number of cpu used (by defualt cpus=4)
`option`	a character used to defined the type of predicted values (by default, type="postcluster", see details)
`encoding`	a character used to defined the type of encoding for the predicted values ("123" or "UHM", see details)
`seed`	a numeric value corresponding to the argument of the function set.seed (by default, seed=NULL).
`...`	further arguments passed to or from other methods

Details

The function 'beta3mix' uses a modified version of the function 'blc' from R packages (RPMM). The function 'predict' proposes three different types of prediction values: meancluster: classification based on averaged cut-off; postcluster: classification based on posterior probabilities; postproba: posterior probability. The argument call 'encoding' offer the possibilities to replace the default encoding ('c(1,2,3)') by an other (e.g. 'c("U","H","M")').

Value

The function 'beta3mix' returns a list of of classes "beta3mix". The list contains parameters and objects representing mixture model fit, including posterior weights and log-likelihood from the function 'blc' from R package RPMM. This list is completed by element related to the classification of the values in the three categories described below:

`idx`	identification of the values used to fit the model
`subcluster`	classification of the values used to fit the model
`lim`	limit values used to defined 'meancluster' classification
`meancluster`	a vector containing the classification of the values based on averaged cut-off (see RPMM package)
`nclass`	number of class
`postw`	a matric containing posterior probabilities
`postcluster`	a vector containing the classification of the values based on maximal posterior probabilities
`call`	the call function

References

Teschendorff AE, Marabita F, Lechner M, Bartlett T, Tegner J, Gomez-Cabrero D, Beck S. A Beta-Mixture Quantile Normalisation method for correcting probe design bias in Illumina Infinium 450k DNA methylation data. Bioinformatics. 2012 Nov 21. Houseman AE and Sc.D. (2012). RPMM: Recursively Partitioned Mixture Model. R package version 1.10. http://CRAN.R-project.org/package=RPMM Koestler DC, Christensen BC, Marsit CJ, Kelsey KT, Houseman EA. (2013) Recursively partitioned mixture model clustering of DNA methylation data using biologically informed correlation structures. Stat Appl Genet Mol Biol. 2013 Mar 5;12(2):225-40.

Examples

```
# + add example
```

`betaUHM` *DNA methylation state or Genoptyping based on 3-beta mixture model*

Description

Classification of Beta-values in using 3-beta mixture distribution and EM algorithm

Usage

```
betaUHM(df,n=10000,initial=c(0.2,0.75),niter=25,tol=1e-4,cpus=4,type="postcluster",moda=c("U","H
```

Arguments

`df`	a sdata.frame containing the beta-value
`n`	a numeric value corresponding to the number of values used to fit the model (10000 by default)
`initial`	initial condition used to defined a priori classification
`niter`	a numeric value corresponding to the maximum number of iteration
`tol`	a numeric value corresponding to the tolerance criteria
`cpus`	a numeric value providing the number of cpu used (by defualt cpus=4)
`type`	a character used to defined the type of output (by default, type="postcluster", see details)
`moda`	a character used to defined the encoding of genotype(by default, moda=c("U","H","M"), see details)
`...`	further arguments passed to or from other methods

Details

For the classification of the values, three different types of prediction values are available: meancluster: classification based on averaged cut-off
postcluster: classification based on posterior probabilities
postproba: posterior probability

Value

return a numeric

References

Teschendorff AE, Marabita F, Lechner M, Bartlett T, Tegner J, Gomez-Cabrero D, Beck S. A Beta-Mixture Quantile Normalisation method for correcting probe design bias in Illumina Infinium 450k DNA methylation data. Bioinformatics. 2012 Nov 21. Houseman AE and Sc.D. (2012). RPMM: Recursively Partitioned Mixture Model. R package version 1.10. http://CRAN.R-project.org/package=RPMM Koestler DC, Christensen BC, Marsit CJ, Kelsey KT, Houseman EA. (2013) Recursively partitioned mixture model clustering of DNA methylation data using biologically informed correlation structures. Stat Appl Genet Mol Biol. 2013 Mar 5;12(2):225-40.

Examples

```
# data(BetaValues)
## three DNA methylation states
# w <- betaUHM(BetaValues,moda=c("U","H","M"))
## deux DNA methylation states
# w <- betaUHM(BetaValues,moda=c("U","M","M"))
## genotyping with A and B alleles
# w <- betaUHM(BetaValues,moda=c("A/A","A/B","B/B"))
## genotyping with numerical encoding
# w <- betaUHM(BetaValues,moda=c(0,1,2))
```

`cghGene` *Extract the CGH information related to a given gene*

Description

The function extracts to object from cghCall class the CGH information for a given gene

Usage

```
cghGene(gene, calls, ...)
```

Arguments

`gene`	a character or a object IRanges
`calls`	an object from cghCall class
`...`	further arguments passed to or from other methods

Details

add details (?)

Value

The function return an object of the class ’cghCall’.

References

Mark A. van de Wiel, Kyung In Kim, Sjoerd J. Vosse, Wessel N. van Wieringen, Saskia M. Wilting and Bauke Ylstra (2007) CGHcall: calling aberrations for array CGH tumor profiles. Bioinformatics, 23,892-894.

`entropy.index` *Computation of the Entropy/Shannon index*

Description

The function computes the Entropy/shannon index.

Usage

```
entropy.index(x,standard=FALSE,...)
```

Arguments

`X`	a vector or a data.frame
`Standard`	a logical value indicating if the value is standardized by maximal Entropy (FALSE by default).
`...`	further arguments passed to or from other methods

Value

The functions returns a numeric or vector containing the entropy values (by row if x is a data.frame).

References

add reference fro Shannon index (?)
hataway (1992)

Examples

```
## example 1
log(4)
x <- c(0.25,0.25,0.25,0.25)
-sum(x*log(x))
entropy.index(x,standard=FALSE)
entropy.index(x,standard=TRUE)
## example 2
x <- rep(4,4)
entropy.index(x,standard=FALSE)
entropy.index(x,standard=TRUE)
```

---

extractCGH — *Extraction of gene or genomic regions into an object 'cghCall'*

Description

The functions extract of 'cghCall' information for a given gene, chromosomes or genomic regions.

Usage

```
extractCGH.gene(x,gene,...)
extractCGH.chr(x,chr,...)
extractCGH.default(x,chr, start,end,winsize=0.0,info=NULL,mode=NULL,...)
```

Arguments

`x`	an object 'cghCall' (output of the function ExpandCGHcall from Rpackage CGHcall).
`chr`	a character corresponding to the chromosome information (e.g. "chr1")
`gene`	a character value corresponding to gene name (Symbol name)
`start`	a numeric value corresponding to the start position of the selected genomic region
`end`	a numeric value corresponding to the end position of the selected genomic region
`info`	a character value describing the genomic region (NULL by default).
`winsize`	a numerical value used to increase the windows of the selected genomic region (in pb)
`mode`	a numeric value to select the selection mode (NULL by default, see details).
`...`	further arguments passed to or from other methods

Details

The estimation is based on mixture models proposed in the R package CGHcall (see functions called 'CGHcall').
The argument called "mode" is used to select the mode of selection: * mode 1 (include): | x————x |
* mode 2 (incomplete overlap): x——|——x |
* mode 3 (incomplete overlap): | x——|——x
* mode 4 (total overlap): x——|—-|–x
By default, the mode is equal to NULL that corresponds to the use of the four selection modes (c(1,2,3,4)).

Value

The functions returns a object 'cghCall' with an addition attributes called "loc" (as location).
This new attribute is a list describing below:

`info`	a character value describing the genomic region, for example gene or chromosome name (NULL by default).
`chr`	a character corresponding to the chromosome information (e.g. "chr1")
`start`	a numeric value corresponding to the start position of the selected genomic region
`end`	a numeric value corresponding to the end position of the selected genomic region
`winsize`	a numerical value used to increase the windows of the selected genomic region (in pb)
`mode`	a numeric value to select the selection mode (NULL by default, see details).

`gene2IR` *Convert a gene symbol, geneloc object or cghCall object into IRanges object*

Description

The functions convert a gene symbol, geneloc object or cghCall object into IRanges object

Usage

```
gene2IR(gene, calls, chrom = NULL, ...)
cghCall2IR(x, chrom, ...)
geneloc2IR(x, ...)
gene2GR(gene, calls, chrom = NULL, ...)
cghCall2GR(x, chrom, ...)
geneloc2GR(x, ...)
```

Arguments

`x`	a character value corresponding to a gene symbol or an object 'geneloc'
`gene`	a character value corresponding to a gene symbol or a object IRanges
`calls`	an cghCall object
`chrom`	an integer value given the chromosome location
`...`	further arguments passed to or from other methods

Details

+ add details (?)

Value

The functions return on IRanges or GRanges object associated with a given gene or an cghCall object.

References

Mark A. van de Wiel, Kyung In Kim, Sjoerd J. Vosse, Wessel N. van Wieringen, Saskia M. Wilting and Bauke Ylstra (2007) CGHcall: calling aberrations for array CGH tumor profiles. Bioinformatics, 23,892-894.

See Also

`GRanges-class`,`IRanges-class`

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (gene, calls, chrom = NULL, ...)
{
    if (!inherits(gene, "IRanges") & is.character(gene)) {
        gene1 <- getGeneLocationR(gene)
        chrom <- gene1$chrom
        chrom <- c(1:24, 23, 24)[match(chrom, c(paste("chr",
            1:24, sep = ""), "chrX", "chrY"))]
        chrom <- as.numeric(chrom)
        if (length(unique(chrom)) > 1)
            warning("multiple chrosome locations!")
        chrom <- unique(chrom)[1]
        wgene <- geneloc2IR(gene1)
    }
    else if (inherits(geneloc2IR, "IRanges") & !is.null(chrom)) {
        wgene <- gene
    }
    else stop("non convenient argument!")
    wcalls <- cghCall2IR(calls, chrom)
    wintersect <- findOverlaps(wcalls, wgene)
    res <- wcalls[attributes(wintersect)$queryHits]
    attr(res, "chrom") <- chrom
    attr(res, "call") <- match.call()
    return(res)
  }
```

getGeneLocationR *Get the location of a vector of gene*

Description

This function provides the location information of a given gene (character) or a set of gene (vector of character) based on the R packages 'org.Hs.eg.db' and 'TxDb.Hsapiens.UCSC.hg19.knownGene'.

Usage

```
getGeneLocationR(genelist, group = TRUE, orderby = "genename",...)
getGeneLocationR2(genelist, group = TRUE, orderby = "genename",onlychr=FALSE,...)
getGeneInfoR(genelist, orderby = "genename",exon=TRUE,cds=TRUE,tx=TRUE,ucsckg=TRUE,...)
getVariantInfoR(chrom,start,end=start,labels=NULL,winsize=0,verbose=FALSE,...)
getVariantInfoR.df(df,control=list(),winsize=0,verbose=FALSE,...)
variant.control(label="rownames",chrom="chrom",start="start",end=start)
```

Arguments

`genelist`	a vector of character
`group`	a logical value (TRUE by default, see details)
`orderby`	a character corresponding to the name of the column used to order the output
`tx`	a logical value (TRUE by default, see details)
`exon`	a logical value (TRUE by default, see details)
`cds`	a logical value (TRUE by default, see details)
`ucsckg`	a logical value (TRUE by default, see details)
`df`	a data.frame containing label,chromosome, end and start position information
`chrom`	a character vector containing chromosome information (e.g. chr11)
`start`	a numeric vector containing start position
`end`	a numeric vector containing end position
`labels`	a character vector containing names information (e.g. rsxxxxx for SNP or cgxxxxxx for DNA methylation probes)
`labels`	a numeric value used to modify the windows size (start-winsize and end-winsize, winsize=0 by default)
`verbose`	a logical value (FALSE by default)
`control`	a list of parameters for controlling the selection of label, chromosome, end and start position information. This is passed to 'variant.control'.
`onlychr`	a logical value for the exclusion of elements with chromosome information containing '_' (by default, onlychr=FALSE).
`...`	further arguments passed to or from other methods

Details

add info on the argument "group"!!!

Value

The function returns an object of the class geneloc corresponding to a data.frame containing gene location information from R packages 'org.Hs.eg.db' and 'TxDb.Hsapiens.UCSC.hg19.knownGene'.

`genename`	the symbol gene
`chrom`	the chromosome location
`txStart`	the physcial location of the starting point
`txEnd`	the physcial location of the ending point
`strand`	the strand of the gene

The data.frame is ordered by genename by default (see argument orderby). ... add more information for the functions getInfoR() and getVariantInfoR()

`getTCGAnames` *TCGA sample name decoding*

Description

The function decodes the names of the samples from TCGA project (optimized for GBM).

Usage

```
getTCGAnames(x, rownames = TRUE, ...)
```

Arguments

`x`	a character
`rownames`	a logical values (by default TRUE). If rownames is equal to TRUE, the rownames of output is the sample names.
`...`	further arguments passed to or from other methods

Details

+ description de l'orgasnisation du code (?) need to check the type of tissue available in the TCGA databases.

Value

a data.frame containing the decoded information related to the name of sample from TCGA project.

References

reference for TCGA project

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (x, rownames = TRUE, ...)
{
    if (is.factor(x))
        x <- as.character(x)
    if (!is.character(x))
        stop("non convenient argument!")
    fdecode <- function(z) {
        w <- list(Name = substring(z, 1, 12), Project = substring(z,
            1, 4), CollectionCenter = substring(z, 6, 7), Patient = substring(z,
            9, 12), SampleType = substring(z, 14, 15), SampleSequence = substring(z,
            16, 16), PortionSequence = substring(z, 18, 19),
            PortionAnalyte = substring(z, 20, 20), PlateID = substring(z,
                22, 25), DataGeneratingCenter = substring(z,
                27, 28))
        w$type <- c("solid tumor", "normal blood", "normal tissue",
            "buccal smear", "cell line")[match(w$SampleType,
            c("01", "10", "11", "12", "20"))]
        return(unlist(w))
    }
    df <- data.frame(do.call("rbind", lapply(x, fdecode)))
    if (rownames)
        row.names(df) <- x
    return(df)
  }
```

`logoffset` *Log link function with an offset*

Description

Computes the log transformation with an offset

Usage

```
logoffset(x, ..., offset = 0, method = "only0")
```

Arguments

`x`	a numeric
`...`	further arguments passed to or from other methods
`offset`	a numeric corresponding to the offset (equal to 0 by default)
`method`	a character

Details

+ description of the argument ’method’

Value

return a numeric

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (x, ..., offset = 0, method = "only0")
{
    switch(method, only0 = ifelse(x == 0, 0 + offset, log(x,
        ...)), all = log(x + offset, ...), stop("non convient method!"))
  }
```

`maxCNAcalls` *Estimation of the CNA for a given region based on Mixture Model at the sample scale.*

Description

set of functions used to estimate the CNA of a given region, chromosome or gene for each individual samples (samples scale estimation). The estimation is based on mixture models proposed in the R package CGHcall.

Usage

```
maxCNAcalls(X,chrom, start,end,winsize=0,mode=NULL,nclass=5,...)
maxCNAcalls.chr(X,chrom,nclass=5,...)
maxCNAcalls.gene(X,gene,nclass=5,...)
## S3 method for class 'CNAcalls'
print(x,nclass=5,...)
```

Arguments

`X`	an object 'cghCall' (output of the function ExpandCGHcall from Rpackage CGHcall).
`x`	an object 'CNAcalls'
`chrom`	a character corresponding to the chromosome information (e.g. "chr1")
`gene`	a character value corresponding to gene name (Symbol name)
`start`	a numeric value corresponding to the start position of the selected genomic region
`end`	a numeric value corresponding to the end position of the selected genomic region
`winsize`	a numerical value used to increase the windows of the selected genomic region (in pb)
`mode`	a numeric value to select the selection mode (NULL by default, see details).
`nclass`	a numeric value to select the number of CNA state (5 by default, see the Function CGHcall from R package CGHcall).
`...`	further arguments passed to or from other methods

Details

The estimation is based on mixture models proposed in the R package CGHcall (see functions called 'CGHcall').
The argument called "mode" is used to select the mode of selection: * mode 1 (include): | x————x |
* mode 2 (incomplete overlap): x——|——x |
* mode 3 (incomplete overlap): | x——|——x
* mode 4 (total overlap): x——|—-|–x
By default, the mode is equal to NULL that corresponds to the use of the four selection modes (c(1,2,3,4)).

Value

The functions returns a list of of classes "CNAcalls":

`cna`	a character vector containing the CNA status for each sample
`n`	a numeric value corresponding to the number of samples
`info`	a charcter value describing the selected genomic regions
`winsize`	a numerical value used to increase the windows of the selected genomic region (in pb)
`location`	a list containning information about the location of the selected region (chr, start and end positions)
`summary`	a numerical vector containing a summarize of the CNA information
`call`	the call function

References

Mark A. van de Wiel, Kyung In Kim, Sjoerd J. Vosse, Wessel N. van Wieringen, Saskia M. Wilting and Bauke Ylstra. CGHcall: calling aberrations for array CGH tumor profiles. Bioinformatics, 23,892-894.

Examples

```
#+ add examples
```

`meth2norm` *Normalization and preparation of the Illumina infinium DNA methylation data*

Description

The function proposes the normalization of the Illumina infinium DNA Methylation data and computation of total intensity (Methylation + Unmethylation). Before the computation of the total intensity, the Unmethylated and Methylated information can be normalized in the same way and time (quantile normalization by default).

Usage

```
meth2norm(x,...)
## Default S3 method:
meth2norm(x,y,method="normalize.quantiles",scaling=FALSE,
chemistry=NULL,rev=FALSE,verbose=TRUE,add.UM=FALSE,...)
## S3 method for class 'MethySet'
meth2norm(x,method="normalize.quantiles",scaling=FALSE,
chemistry=NULL,rev=FALSE,verbose=TRUE,add.UM=FALSE,...)
```

Arguments

`x`	an object 'MethySet' from preprocess functions of package 'minfi' or a matrix or data.frame containing probes for unmethylation
`y`	a matrix or data.frame containing probes for methylation (if x is a matrix or data.frame containing probes for unmethylation).
`method`	a character corresponding to normalization method (by default 'normalize.quantiles' from package preprocessCore)
`scaling`	a logical value indicating if the scaling factor procedure is used to correct the chemistry effect (by default 'FALSE')
`...`	further arguments passed to or from other methods

Details

The unmethylated and methylated tables are normalized in the same tables after concatenation of them (by default, we propose the use of quantile normalization). The total intensity is obtained by sum of the normalized unmethylated table and normalized methylated table.

Value

The function returns a list of three normalized matrix

`T`	an object 'matrix' containing the total intensity
`U`	an object 'matrix' containing the normalized unmethylation intensity
`M`	an object 'matrix' containing the normalized unmethylation intensity
`colnames`	a vector of character containing the sample names
`rownames`	a vector of character containing the probe names
`call`	generally 'match.call()'

References

Staaf, Johan and Vallon-Christersson, Johan and Lindgren, David and Juliusson, Gunnar and Rosenquist, Richard and Hoglund, Mattias and Borg, Ake and Ringner, Markus (2008) Normalization of Illumina Infinium whole-genome SNP data improves copy number estimates and allelic intensity ratios,BMC Bioinformatics, 9:409 Louhimo, R. and Hautaniemi, S. (2011) CNAmet: an R package for integration of copy number, expression and methylation data Bioinformatics 27(6):887-888.

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (...)
{
    UseMethod("meth2norm")
  }
```

`mixCNAcalls` *Estimation of the CNA for a given region based on Mixture Model at the sample scale.*

Description

set of functions used to estimate the CNA of a given region, chromosome or gene for each individual samples (samples scale estimation). The estimation is based on mixture models proposed in the R package CGHcall.

Usage

```
mixCNAcalls(X,chrom, start,end,winsize=0,mode=NULL,nclass=5,...)
mixCNAcalls.chr(X,chrom,nclass=5,...)
mixCNAcalls.gene(X,gene,nclass=5,...)
## S3 method for class 'mixCNAcalls'
print(x,nclass=5,...)
```

Arguments

`X`	an object ’cghCall’ (output of the function ExpandCGHcall from Rpackage CGHcall).
`x`	an object ’CNAcalls’
`chrom`	a character corresponding to the chromosome information (e.g. "chr1")
`gene`	a character value corresponding to gene name (Symbol name)
`start`	a numeric value corresponding to the start position of the selected genomic region
`end`	a numeric value corresponding to the end position of the selected genomic region
`winsize`	a numerical value used to increase the windows of the selected genomic region (in pb)
`nclass`	a numeric value to select the number of CNA state (5 by default, see the Function CGHcall from R package CGHcall).
`...`	further arguments passed to or from other methods

Details

The estimation is based on mixture models proposed in the R package CGHcall (see functions called ’CGHcall’).
explain the definition of the limits

Value

The functions returns a list of of classes "mixCNAcalls":

`cna`	a character vector containing the CNA status for each sample
`n`	a numeric value corresponding to the number of samples
`info`	a charcter value describing the selected genomic regions
`winsize`	a numerical value used to increase the windows of the selected genomic region (in pb)
`location`	a list containning information about the location of the selected region (chr, start and end positions)
`summary`	a numerical vector containing a summarize of the CNA information
`call`	the call function

References

Mark A. van de Wiel, Kyung In Kim, Sjoerd J. Vosse, Wessel N. van Wieringen, Saskia M. Wilting and Bauke Ylstra. CGHcall: calling aberrations for array CGH tumor profiles. Bioinformatics, 23,892-894.

Examples

```
#+ add examples
```

`pathway.test` *Pathway test by hyperfeometric test*

Description

This function provide a pathway test by hyperfeometric test

Usage

```
pathway.test(DEgene,pathway,p.method="bonferroni",...)
pathway.mctest(DEgene,pathway,p.method="bonferroni",cpus=2,...)
```

Arguments

`DEgene`	a vector of binary values. the value is 0 for unselected genes and 1 for selected genes. the names of the vector can be EntrezID.
`pathway`	list of genes (same type/id used to name DEgene element) from PID, GO, Kegg, etc ...
`p.method`	see the argument 'method' related to the R functions p.adjust
`cpus`	an integer corresponding to the number of cpus used for computation (see package snowfall)
`...`	further arguments passed to or from other methods

Details

+ details ?! fisher.test(matrix(c(a,c,b,d),2,2),alternative="greater") phyper(a-1,a+b,c+d,a+c,lower.tail=FALSE)

Value

The function returns a data.frame containing the information related to pathway test.

`n`	total number of gene
`in1`	number of selected gene in pathway
`in0`	number of unselected gene in pathway
`out1`	number of unselected gene not in pathway
`out0`	number of unselected gene not in pathway
`oddsratio`	estimate odds ratio
`p.raw`	p.value from hypergeometric test
`p.adj`	adjusted p.value (see argument 'p.method')

+

### Example with GO with R packages gage # per type MF, CC or BP gomf <- go.hs$go.sets[go.hs$go.subs$MF] degene1 <- sample(c(0,1),length(unique(unlist(gomf)[1:10000])),replace=TRUE) names(degene1) <- unique(unlist(gomf)[1:10000]) table(degene1)

# Go analysis require(gage) go.hs <- go.gsets(species="human") names(go.hs) gogs <- go.hs$go.sets

ted_go <- pathway.test(degene1,gomf[1:8]) ted_go

### Example with PID and NCI from R package CePa require(CePa) data(PID.db) names(PID.db) ted_nci <- pathway.test(degene1,PID.db$NCI$pathList[1:8]) ted_nci

---

`prep.annot450k` *Preparation of annotation for Infinium HM-450K*

Description

Preparation of annotation file

Usage

```
prep.annot450k(x, snp.rm = TRUE, sex.rm = TRUE,in27k=FALSE,chemistry=NULL)
```

Arguments

`x`	a data.frame containing the annotation of Infinium HM-450K platform
`snp.rm`	a logical value. If TRUE, probes located on snp locations are excluded (by default TRUE).
`sex.rm`	a logical value. If TRUE, probes located on sex chromosomes are excluded (by default TRUE).
`in27k`	a logical value. If TRUE, probes common to HM-27K and HM-450K platforms (by default FALSE).
`chemistry`	a character value. Selection of for chemistry type I ("I"), for chemistry type II ("II") or the both chemistry type (by default NULL).

Details

this function is optimized to works on annotation files from Illumina compagny:
- HumanMethylation450_15017482_v.1.1_ForExcel.csv - humanmethylation450_15017482_v1-2.csv

Value

a data.frame containing the annotation of Infinium HM-450K platform filtered for SNP and Sex information

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (x, snp.rm = TRUE, sex.rm = TRUE)
{
    if (sex.rm)
        x <- x[!is.element(x$CHR, c("X", "Y")), ]
    if (snp.rm)
        x <- x[is.element(x$Probe_SNPs, c("")) & is.element(x$Probe_SNPs_10,
            c("")), ]
   x$CHR2 <- factor(as.character(x$CHR), levels = as.character(sort(as.numeric(levels(x$CHR)))))
    levels(x$CHR2)
    x[order(x$CHR2, x$MAPINFO), ]
  }
```

REout *Remove Bracket, sapce or zero values contained in a character object*

Description

The functions remove some specific symbols (such as bracket, space or zero) in a character object.

Usage

```
REbracketout(x, replaceby = "", ...)
REspaceout(x, replaceby = "", ...)
REzeroout(x, replaceby = "", ...)
```

Arguments

x	a vector of character
replaceby	a character value replacing bracket, space or zero value contained in x.
...	further arguments passed to or from other methods

Value

return a vector of character (string) where bracket, space or zero symbols was removed.

See Also

gsub

`searchGene` *search "gene" containing in a character vector*

Description

This function offers the possibility to search gene (character pattern) containing in a character vector separted by a given separator (semi-colon by default).

Usage

```
searchGene(x,gene,sep=";",type="logical")
```

Arguments

`x`	a character vector
`gene`	the gene (character pattern) of interest
`sep`	separator between the gene (by default, sep=";")
`type`	a character indicating the type of values returned by the function. By default the function return a logical vector (type="logical"). The type "numeric" and "value" return position and value respectively.

Value

The functiuon returns a vector of logical (type="logical"), numeric (type="numeric") or character (type="value") depending of the selected type.

Examples

```
genelist <-c("MGMT","MGMT;MGMT","MGMT;EGFR","HOXA1;EGFR;HOXA2","EGFR;MGMT",NA)
searchGene(genelist,"EGFR",sep=";",type="logical")
searchGene(genelist,"EGFR",sep=";",type="numeric")
searchGene(genelist,"EGFR",sep=";",type="value")
```

`seg2tab` *Convert Segmentation data of an object 'cghSeg' or 'cghCall' in a matrix (or list of matrix)*

Description

The function converts segmentation data ("chrom","start","end","num.mark","seg.mean") of an object 'cghSeg' or 'cghCall' in a matrix (or list of matrix)

Usage

```
seg2tab(x,...)
```

Arguments

`x`	an object 'cghSeg' or 'cghCall'
`...`	further arguments passed to or from other methods

Details

+ details ?!

Value

The function returns an matrix of a list of matrix (if there are several sample) containing the segmentation information obtained by the the function DNAcopy from the R package DNAcopy.

`chrom`	the chromosome within the sample
`start`	the starting map location of the segment
`end`	the ending map location of the segment
`num.mark`	the number of markers in the segment
`seg.mean`	the segment mean

References

Mark A. van de Wiel, Kyung In Kim, Sjoerd J. Vosse, Wessel N. van Wieringen, Saskia M. Wilting and Bauke Ylstra (2007) CGHcall: calling aberrations for array CGH tumor profiles. Bioinformatics, 23,892-894.

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
```

UHM — *Three-Mixture models for DNA methylation genotyping*

Description

These functions fitting models for DNA methylation genotyping based on M-values (gamma or normal version) or Beta-values (beta version).

Usage

```
normalUHM(df,cpus=3,verbose=FALSE,...)
gammaUHM(df,lim=c(0.95,0.95),cpus=3,...)
```

Arguments

`df`	a data.frame containing beta or M-values
`cpus`	an integer corresponding to the number of cpus used for computation (see package snowfall)
`lim`	a numeric vector containing the limit used to defined the DNA methylation genotype
`verbose`	logical, if TRUE information are printed during each step of the algorithm
`...`	further arguments passed to or from other methods

Details

+ details ?!

Value

add retunr

References

add references lumi mixture models

Examples

```
# add exemples
```

Index