Package 'Starr'

April 5, 2014

Version 1.18.1

Date 2009-10-12

Title Simple tiling array analysis of Affymetrix ChIP-chip data

Author Benedikt Zacher, Johannes Soeding, Pei Fen Kuan, Matthias Siebert, Achim Tresch

Maintainer Benedikt Zacher <zacher@lmb.uni-muenchen.de>

Depends Ringo, affy, affxparser, lattice

Imports pspline, MASS, zlibbioc

Description Starr facilitates the analysis of ChIP-chip data, in particular that of Affymetrix tiling arrays. The package provides functions for data import, quality assessment, data visualization and exploration. Furthermore, it includes high-level analysis features like association of ChIP signals with annotated features, correlation analysis of ChIP signals and other genomic data (e.g. gene expression), peak-finding with the CMARRT algorithm and comparative display of multiple clusters of ChIP-profiles. It uses the basic Bioconductor classes ExpressionSet and probeAnno for maximum compatibility with other software on Bioconductor. All functions from Starr can be used to investigate preprocessed data from the Ringo package, and vice versa. An important novel tool is the the automated generation of correct, up-to-date microarray probe annotation (bpmap) files, which relies on an efficient mapping of short sequences (e.g. the probe sequences on a microarray) to an arbitrary genome.

License Artistic-2.0

biocViews Microarray,OneChannel,DataImport,QualityControl,Preprocessing,ChIPchip

LazyLoad yes

R topics documented:

`bpmapToProbeAnno` *Creating a probeAnno object*

Description

This function allows the user to create a probeAnno environment that holds the mapping between probes on the array and their genomic match position(s). The function takes an Affymetrix bpmap file as input.

Usage

```
bpmapToProbeAnno(bpmap, verbose=T, uniqueSeq=T)
```

Arguments

bpmap	Either a list, created by the function readBpmap() from the affy package. Or a path to the bpmap file.
verbose	should the progress be printed out?
uniqueSeq	If TRUE, probes sequences that occur more than once on the chip (and consequently match several positions on the genome) are set to 1 in the probeAnno object. Unique probes are set to 0. If false, all probes are set to 0. To identify all unique and multiple matching probes, a remapping of the probes to the genome is recommended.

Author(s)

Benedikt Zacher <zacher@lmb.uni-muenchen.de>

Examples

```
##
# dataPath <- system.file("extdata", package="Starr")
# bpmapChr1 <- readBpmap(file.path(dataPath, "Scerevisiae_tlg_chr1.bpmap"))

# probeAnnoChr1 <- bpmapToProbeAnno(bpmapChr1)
```

---

cmarrt.ma *Compute moving average statistics by incorporating the correlation structure*

Description

This function extends the moving average approach by incorporating the correlation structure. It also outputs the p-values of the standardized moving average statistics under the Gaussian approximation.

Usage

```
cmarrt.ma(eSet, probeAnno, chr=NULL, M=NULL,frag.length,window.opt=fixed.probe)
```

Arguments

eSet	ExpressionSet containing the normalized ratio
probeAnno	probeAnno object with mapping
chr	which chromosome should be analysed? If chr==NULL, all chromosome in the probeAnno object are taken.
M	rough estimate of the percentage of bound probes. If unknown, leave it NULL.
frag.length	average fragment length from sonication.
window.opt	option for sliding window, either "fixed.probe" or "fixed.gen.dist". Default is 'fixed.probe'.

Details

Computation using `window.opt = "fixed.probe"` calculates the moving average statistics within a fixed number of probes and is more efficient. Use this option if the tiling array is regular with approximately constant resolution. `window.opt="fixed.gen.dist"` computes the moving average statistics over a fixed genomic distance.

Value

`data.sort`	datafile sorted by genomic position.
`ma`	unstandardized moving average(MA) statistics.
`z.cmarrt`	standardized MA under correlation structure.
`z.indep`	standardized MA under independence (ignoring correlation structure).
`pv.cmarrt`	p-values of probes under correlation.
`pv.indep`	p-values of probes under independence (ignoring correlation structure).

Note

The p-values are obtained under the Gaussian approximation. Therefore, it is important to check the normal quantile-quantile plot if the Gaussian approximation is valid. The function also outputs the computation under independence (ignoring the correlation structure) for comparisons.

Author(s)

Pei Fen Kuan, Adam Hinz

References

P.F. Kuan, H. Chun, S. Keles (2008). CMARRT: A tool for the analysiz of ChIP-chip data from tiling arrays by incorporating the correlation structure. *Pacific Symposium of Biocomputing* **13**:515-526.

See Also

`plotcmarrt`,`cmarrt.peak`

Examples

```
# dataPath <- system.file("extdata", package="Starr")
# bpmapChr1 <- readBpmap(file.path(dataPath, "Scerevisiae_tlg_chr1.bpmap"))

# cels <- c(file.path(dataPath,"Rpb3_IP_chr1.cel"), file.path(dataPath,"wt_IP_chr1.cel"),
#  file.path(dataPath,"Rpb3_IP2_chr1.cel"))
# names <- c("rpb3_1", "wt_1","rpb3_2")
# type <- c("IP", "CONTROL", "IP")
# rpb3Chr1 <- readCelFile(bpmapChr1, cels, names, type, featureData=TRUE, log.it=TRUE)

# ips <- rpb3Chr1$type == "IP"
# controls <- rpb3Chr1$type == "CONTROL"
```

```
# rpb3_rankpercentile <- normalize.Probes(rpb3Chr1, method="rankpercentile")
# description <- c("Rpb3vsWT")
# rpb3_rankpercentile_ratio <- getRatio(rpb3_rankpercentile, ips, controls, description, fkt=median, featureData=

# probeAnnoChr1 <- bpmapToProbeAnno(bpmapChr1)
# peaks <- cmarrt.ma(rpb3_rankpercentile_ratio, probeAnnoChr1, chr=NULL, M=NULL,250,window.opt=fixed.probe)
```

`cmarrt.peak` *Obtain bound regions for a given error rate control*

Description

Obtain bound regions under a given error rate control using correction method from `p.adjust`.

Usage

```
cmarrt.peak(cmarrt.ma, alpha, method, minrun, asCherList=FALSE)
```

Arguments

`cmarrt.ma`	output object from `cmarrt.ma`.
`alpha`	error rate control for declaring bound region.
`method`	correction method inherited from `p.adjust`.
`minrun`	minimum number of probes to be called a bound region.
`asCherList`	If TRUE, result is returned as class cherList. See Ringo, for further description.

Details

The function returns two objects, `cmarrt.bound` and `indep.bound`. Each object is a list of bound regions which can be accessed by `$chr` (chromosome), `$peak.start` (start coordinate of each bound region), `$peak.stop` (stop coordinate of each bound region), `$n.probe` (number of probes within each bound region), `$min.pv` (minimum p-values of each bound region), `$ave.pv` (average p-values of each bound region).

Value

`cmarrt.bound`	list of bound regions obtained under correlation structure.
`indep.bound`	list of bound regions obtained under independence (ignoring correlation).

Note

The list of bound regions obtained under independence (ignoring the correlation structure) is for comparison. It is not recommended to use this list for downstream analysis.

Author(s)

Pei Fen Kuan, Adam Hinz

References

P.F. Kuan, H. Chun, S. Keles (2008). CMARRT: A tool for the analysiz of ChIP-chip data from tiling arrays by incorporating the correlation structure. *Pacific Symposium of Biocomputing***13**:515-526.

See Also

cmarrt.ma,p.adjust

Examples

```
# dataPath <- system.file("extdata", package="Starr")
# bpmapChr1 <- readBpmap(file.path(dataPath, "Scerevisiae_tlg_chr1.bpmap"))

# cels <- c(file.path(dataPath,"Rpb3_IP_chr1.cel"), file.path(dataPath,"wt_IP_chr1.cel"),
#  file.path(dataPath,"Rpb3_IP2_chr1.cel"))
# names <- c("rpb3_1", "wt_1","rpb3_2")
# type <- c("IP", "CONTROL", "IP")
# rpb3Chr1 <- readCelFile(bpmapChr1, cels, names, type, featureData=TRUE, log.it=TRUE)

# ips <- rpb3Chr1$type == "IP"
# controls <- rpb3Chr1$type == "CONTROL"

# rpb3_rankpercentile <- normalize.Probes(rpb3Chr1, method="rankpercentile")
# description <- c("Rpb3vsWT")
# rpb3_rankpercentile_ratio <- getRatio(rpb3_rankpercentile, ips, controls, description, fkt=median, featureData=

# probeAnnoChr1 <- bpmapToProbeAnno(bpmapChr1)
# peaks <- cmarrt.ma(rpb3_rankpercentile_ratio, probeAnnoChr1, chr=NULL, M=NULL,250,window.opt=fixed.probe)
# peaklist <- cmarrt.peak(peaks)
```

`correlationPlot` *correlation of ChIP signals to other data*

Description

`correlationPlot` The correlationPlot can be used to vizualize e.g. the correlation between the mean binding intensity of specific regions around annotated features and gene expression. The regions around the annotated features, that should be analyzed are defined in a data frame. Each row represents one region. In the upper panel of the plot, the correlation is plotted in a barplot. In the lower panel, the annotated feature and the regions defined in the data frame are shown.

Usage

```
correlationPlot(regions, labels=c("start", "stop"), ...)
```

Arguments

regions	a data frame, containing four columns. Every row defines one region to be analyzed and is plotted in the lower panel. pos=start, upstream=500 and downstream=500 mean characterize the region 500 bp upstream and downstream around the start of the feature. The pos columns is a character with values out of c("start", "region", "end"). upstream and downstream ar integers, indicating how many bases upstream and downstream from the specified position in the feature are included. level is an integer, that says at which level the rectangle in the lower device should be plotted. The numeration goes from the bottom to the ceiling. cor is the correlation of the region, which is plotted in the upper panel.
labels	a character vector which holds the names of the borders of the annotated region. (e.g. c("TSS", "TTS") for transcripts)
...	parameters, that are passed to barplot (for plotting the upper panel)

Author(s)

Benedikt Zacher <zacher@lmb.uni-muenchen.de>

See Also

barplot

Examples

```
## Constructing an example data frame
pos <- c("start", "start", "start", "region", "region","region","region", "stop","stop","stop")
upstream <- c(500, 0, 250, 0, 0, 500, 500, 500, 0, 250)
downstream <- c(0, 500, 250, 0, 500, 0, 500, 0, 500, 250)
level <- c(1, 1, 2, 3, 4, 5, 6, 1, 1, 2)
cor <- seq(-1,1, length=10)
info <- data.frame(pos=pos, upstream=upstream, downstream=downstream, level=level, cor=cor, stringsAsFactors=FALS
rownames(info) <- letters[1:10]

## Generate plot
correlationPlot(info)
```

---

densityscatter *Compute density of a scatterplot*

Description

A 2d density is computed by kde2D.

Usage

```
densityscatter(x,y,pch=19,cex=1,ncol=30,grid=100,palette="heat", add=F,...)
```

Arguments

x	x coordinate of data
y	y coordinate of data
pch	type of point
cex	A numerical value giving the amount by which plotting text and symbols should be magnified relative to the default
grid	Number of grid points in each direction
ncol	number of colors
palette	color palette to choose
add	should data points be added to an exisiting plot?
...	parameters passed to plot or points

Author(s)

Benedikt Zacher <zacher@lmb.uni-muenchen.de>

See Also

kde2dplot

Examples

```
##
points = 10^4
x <- rnorm(points/2)
x = c(x,x+2.5)
y <- x + rnorm(points,sd=0.8)
x = sign(x)*abs(x)^1.3
densityscatter(x,y)
```

---

expressionByFeature *Getting expression value by feature from an ExpressionSet*

Description

This function gets the expression of a specified feature (e.g. orf, gene) from an ExpressionSet.

Usage

```
expressionByFeature(eSet, fkt, method="median")
```

Arguments

eSet	An ExpressionSet, containing the normalized expression values
fkt	Function to convert the featureNames (e.g. affy IDs) of eSet to the required features (e.g. ORFs)
method	If one feature (e.g. ORF) has more than one feature (e.g. affy ID) on the chip, the mean/median over the intensities is taken

Author(s)

Benedikt Zacher <zacher@lmb.uni-muenchen.de>

See Also

mget

---

filterGenes *Filter Features/Genes*

Description

This fucntion filters genes and other annotated features with respect to length, ovelaps and distance to other features.

Usage

```
filterGenes(gffAnno, distance_us=500, distance_ds=500, minLength=-Inf, maxLength=Inf)
```

Arguments

gffAnno	a data frame containing the annotation
distance_us	how many basepairs upstream to the feature should not overlap with other features.
distance_ds	how many basepairs downstream to the feature should not overlap with other features.
minLength	minimal length of the feature
maxLength	maximal length of the feature

Value

a character vector with the names of the features, that passed the filter.

Author(s)

Benedikt Zacher <zacher@lmb.uni-muenchen.de>

Examples

```
##
# dataPath <- system.file("extdata", package="Starr")
# transcriptAnno <- read.gffAnno(file.path(dataPath, "transcriptAnno.gff"), feature="transcript")
# filtered_transcripts <- filterGenes(transcriptAnno, distance_us = 0, distance_ds = 0, minLength = 1000)
```

---

getMeans *Get mean ChIP-signal over annotated features*

Description

getMeans calculates the mean ChIP-signal over annotated features

Usage

```
getMeans(eSet, probeAnno, geneAnno, regions)
```

Arguments

eSet	an ExpressionSet
probeAnno	a probeAnno object for the given ExpressionSet
geneAnno	a data frame containing the annotation of the features of interest
regions	a data frame, containing four columns. The pos columns is a character with values out of c("start", "region", "end"). upstream and downstream ar integers, indicating how many bases upstream and downstream from the specified position in the feature are included. level is an integer, that says at which level the rectangle in the lower device should be plotted. The numeration goes from the bottom to the ceiling. cor is the correlation of the region, which is plotted in the upper panel.

Value

a list. Each entry contains the mean signals over the specified regions (in the regions data frame) of all features in geneAnno.

Author(s)

Benedikt Zacher <zacher@lmb.uni-muenchen.de>

See Also

getProfiles

Examples

```
##
# dataPath <- system.file("extdata", package="Starr")
# bpmapChr1 <- readBpmap(file.path(dataPath, "Scerevisiae_tlg_chr1.bpmap"))

# cels <- c(file.path(dataPath,"Rpb3_IP_chr1.cel"), file.path(dataPath,"wt_IP_chr1.cel"),
#  file.path(dataPath,"Rpb3_IP2_chr1.cel"))
# names <- c("rpb3_1", "wt_1","rpb3_2")
# type <- c("IP", "CONTROL", "IP")
# rpb3Chr1 <- readCelFile(bpmapChr1, cels, names, type, featureData=TRUE, log.it=TRUE)

# ips <- rpb3Chr1$type == "IP"
# controls <- rpb3Chr1$type == "CONTROL"

# rpb3_rankpercentile <- normalize.Probes(rpb3Chr1, method="rankpercentile")
# description <- c("Rpb3vsWT")
# rpb3_rankpercentile_ratio <- getRatio(rpb3_rankpercentile, ips, controls, description, fkt=median, featureData=

# probeAnnoChr1 <- bpmapToProbeAnno(bpmapChr1)

# transcriptAnno <- read.gffAnno(file.path(dataPath, "transcriptAnno.gff"), feature="transcript")
# filtered_orfs <- filterGenes(transcriptAnno, distance_us = 0, distance_ds = 0, minLength = 1000)

# pos <- c("start", "start", "start", "region", "region","region","region", "stop","stop","stop")
# upstream <- c(500, 0, 250, 0, 0, 500, 500, 500, 0, 250)
# downstream <- c(0, 500, 250, 0, 500, 0, 500, 0, 500, 250)
# info <- data.frame(pos=pos, upstream=upstream, downstream=downstream, stringsAsFactors=FALSE)
# means_rpb3 <- getMeans(rpb3_rankpercentile_ratio, probeAnnoChr1, transcriptAnno[which(transcriptAnno$name %in%
```

`getProfiles` *Get profiles of ChIP-signal over annotated features*

Description

This function associates the measured ChIP signals to annotated features and stores the profile of each feature in a list. Each profile is divided in three parts. The first entry is "upstream", which saves the signal upstream of start. Then follows "region", which is from start to end and then "downstream", which stores the signals downstream of end.

Usage

```
getProfiles(eSet, probeAnno, gffAnno, upstream, downstream, feature="ORF", borderNames, method, sameLe
```

Arguments

`eSet`	an ExpressionSet, containing on sample.
`probeAnno`	a probeAnno object for the given ExpressionSet
`gffAnno`	a data frame containing the annotation of the features of interest

upstream	how many basepairs upstream of the feature start (feature start on the crick strand is end in gffAnno) should be taken.
downstream	how many basepairs downtream of the feature start (feature end on the crick strand is start in gffAnno) should be taken.
feature	name of the features (e.g. ORF, transcript, rRNA, ...)
borderNames	names of the borders, flaking the feature (e.g. c("start", "stop"))
method	Two methods are available. "middle", just takes the middle position of each probe and its corresponding value. This method should be used if the whole genome is tiled in an high resolution. "basewise" calculates for each base the mean of all probes overlapping with this position.
fill	if "middle" is chosen the distance of the taken values equals the probe spacing on the chip. To avoid errors, because of regions lacking of probes, one can fill up these regions with NAs.
distance	if method "middle" and fill==TRUE are chosen, distance is the max distance of no value occuring before filling in one NA.
spacing	probe spacing on the chip. Only used for filling up with NAs in method "middle".
sameLength	if method "middle" is chosen it can occur that the length of the upstream/downstream region vary a little. If sameLength==TRUE, upstream/downstream regions get all the same length.

Value

a list with the following entries

ID	the ID/name of the sample
upstream	number of basepairs, taken upstream of the feature
downstream	number of basepairs, taken upstream of the feature
method	method used
borderNames	names of the borders
feature	feature type (e.g. "ORF")
profile	a list which contains all profiles of the features in the gffAnno. Each entry consists of a list with the elements "upstream", "region", "downstream".

Author(s)

Benedikt Zacher <zacher@lmb.uni-muenchen.de>

See Also

fill,fillNA,mapFeatures,getIntensities,getFeature, fill,getProfilesByBase

Examples

```
##
# dataPath <- system.file("extdata", package="Starr")
# bpmapChr1 <- readBpmap(file.path(dataPath, "Scerevisiae_tlg_chr1.bpmap"))

# cels <- c(file.path(dataPath,"Rpb3_IP_chr1.cel"), file.path(dataPath,"wt_IP_chr1.cel"),
#  file.path(dataPath,"Rpb3_IP2_chr1.cel"))
# names <- c("rpb3_1", "wt_1","rpb3_2")
# type <- c("IP", "CONTROL", "IP")
# rpb3Chr1 <- readCelFile(bpmapChr1, cels, names, type, featureData=TRUE, log.it=TRUE)

# ips <- rpb3Chr1$type == "IP"
# controls <- rpb3Chr1$type == "CONTROL"

# rpb3_rankpercentile <- normalize.Probes(rpb3Chr1, method="rankpercentile")
# description <- c("Rpb3vsWT")
# rpb3_rankpercentile_ratio <- getRatio(rpb3_rankpercentile, ips, controls, description, fkt=median, featureData=

# probeAnnoChr1 <- bpmapToProbeAnno(bpmapChr1)
# transcriptAnno <- read.gffAnno(file.path(dataPath, "transcriptAnno.gff"), feature="transcript")

# profile <- getProfiles(rpb3_rankpercentile_ratio, probeAnnoChr1, transcriptAnno, 500, 500, feature="transcript"
```

`getRatio` *Building ratio over experiments*

Description

This function calculates the ratio over experiments.

Usage

```
getRatio(eSet, ip, control, description, fkt=median, featureData=F)
```

Arguments

`eSet`	An ExpressionSet, containing the logged raw intensities
`ip`	a boolean or integer vector, that indicate, which columns in the matrix are IP experiments
`control`	a boolean or integer vector, that indicate, which columns in the matrix are CONTROL or REFERENCE experiments
`description`	description of the new data (e.g. IPvsCONTROL)
`fkt`	mean or median to calculate the averaged intensity over replicates
`featureData`	if TRUE, featureData is added to the new ExpressionSet

Author(s)

Benedikt Zacher <zacher@lmb.uni-muenchen.de>

Examples

```
##
# dataPath <- system.file("extdata", package="Starr")
# bpmapChr1 <- readBpmap(file.path(dataPath, "Scerevisiae_tlg_chr1.bpmap"))

# cels <- c(file.path(dataPath,"Rpb3_IP_chr1.cel"), file.path(dataPath,"wt_IP_chr1.cel"),
#  file.path(dataPath,"Rpb3_IP2_chr1.cel"))
# names <- c("rpb3_1", "wt_1","rpb3_2")
# type <- c("IP", "CONTROL", "IP")
# rpb3Chr1 <- readCelFile(bpmapChr1, cels, names, type, featureData=TRUE, log.it=TRUE)

# ips <- rpb3Chr1$type == "IP"
# controls <- rpb3Chr1$type == "CONTROL"

# rpb3_rankpercentile <- normalize.Probes(rpb3Chr1, method="rankpercentile")
# description <- c("Rpb3vsWT")
# rpb3_rankpercentile_ratio <- getRatio(rpb3_rankpercentile, ips, controls, description, fkt=median, featureData=
```

---

heatmapplot *heatmapplot*

Description

Heatmap representation of binding profiles

Usage

```
heatmapplot(profiles, colpal=c("black","dark blue","dark green", "green","gold", "yellow"), abl=NULL,
```

Arguments

profiles	a list of profiles returned by getProfiles(). Features must have same lengths.
colpal	color palette for intensity coding
abl	positions of vertical lines that are added to the panel
subset	subset of genes in the list that should be plotted

Author(s)

Benedikt Zacher <zacher@lmb.uni-muenchen.de>

list2matrix *Convert profile list to matrix*

Description

This function converts the list of profiles generated by the getProfiles function to a matrix, if all entries have the same length.

Usage

```
list2matrix(profiles)
```

Arguments

profiles	a list, generated by the getProfiles

Value

a list with with a matrix at the entry profile.

Author(s)

Benedikt Zacher <zacher@lmb.uni-muenchen.de>

makeProbeAnno *Creating a probeAnno object*

Description

Creates a probeAnno object (package: Ringo) from a given Affymetrix bpmap file or a Nimblegen POS file. The posToProbeAnno function from the Ringo package is called to build the object.

Usage

```
makeProbeAnno(posFile=NULL, bpmap=NULL, probeIDAsStrings=F)
```

Arguments

posFile	path to the POS file
bpmap	Either a list, created by the function readBpmap() from the affy package, or a path to the bpmap file.
probeIDAsStrings	should the mapping of the probes to the rows in the assayData be integers or characters.

Author(s)

Benedikt Zacher <zacher@lmb.uni-muenchen.de>

See Also

posToProbeAnno,readBpmap

---

makeSplines *Fit splines to profiles*

Description

This function uses the pspline package to fit spilnes to each entry in a list of profiles.

Usage

```
makeSplines(profiles, df=1000)
```

Arguments

profiles	a list as it is created by the getProfiles package.
df	the degree of freedom of the fit

Value

a list as it is created by the getProfiles function.

Author(s)

Benedikt Zacher <zacher@lmb.uni-muenchen.de>

See Also

smooth.Pspline, predict.smooth.Pspline

Examples

```
##
# dataPath <- system.file("extdata", package="Starr")
# bpmapChr1 <- readBpmap(file.path(dataPath, "Scerevisiae_tlg_chr1.bpmap"))

# cels <- c(file.path(dataPath,"Rpb3_IP_chr1.cel"), file.path(dataPath,"wt_IP_chr1.cel"),
#  file.path(dataPath,"Rpb3_IP2_chr1.cel"))
# names <- c("rpb3_1", "wt_1","rpb3_2")
# type <- c("IP", "CONTROL", "IP")
# rpb3Chr1 <- readCelFile(bpmapChr1, cels, names, type, featureData=TRUE, log.it=TRUE)

# ips <- rpb3Chr1$type == "IP"
```

```
# controls <- rpb3Chr1$type == "CONTROL"

# rpb3_rankpercentile <- normalize.Probes(rpb3Chr1, method="rankpercentile")
# description <- c("Rpb3vsWT")
# rpb3_rankpercentile_ratio <- getRatio(rpb3_rankpercentile, ips, controls, description, fkt=median, featureData=

# probeAnnoChr1 <- bpmapToProbeAnno(bpmapChr1)
# transcriptAnno <- read.gffAnno(file.path(dataPath, "transcriptAnno.gff"), feature="transcript")

# profile <- getProfiles(rpb3_rankpercentile_ratio, probeAnnoChr1, transcriptAnno, 500, 500, feature="transcript"
# profile_splines <- makeSplines(profile)
```

`normalize.Probes` *Normalization of probes*

Description

Normalization of probe intensities with a given method.

Usage

```
normalize.Probes(eSet, method=NULL, ratio=FALSE, ip, control, description, fkt=median, featureData=FAL
```

Arguments

`eSet`	An ExpressionSet, containing the logged raw intensities
`method`	character string specifying the normalization method to be used. Choices are "none", "scale", "quantile", "Aquantile", "Gquantile", "Rquantile", "Tquantile", "vsn", "rankpercentile", "loess", "substract".
`ratio`	if TRUE, the ratios are calcualted.
`ip`	a boolean vector, indicating which sample are IP experiments
`control`	a boolean vector, indicating which sample are CONTROL experiments
`description`	description of the normalized data
`fkt`	function to chose for averaging over replicates
`featureData`	should the featureData of eSet be passed to the new ExpressionSet?
`targets`	vector, factor or matrix of length twice the number of arrays, used to indicate target groups if method="Tquantile"
`arrays`	Subset of experiments (colnames in ExpressionSet) in the ExpressionSet, that are supposed to be normalized seperately.
`...`	arguments, that should be passed to the normalization methods.

Details

The procedure calls different functions from this package or from affy and limma, depending on the method.

none Calls `normalizeWithinArrays` with `method="none"` from package limma.

scale Calls `normalizeWithinArrays` with `method="scale"` from package limma.

quantile Calls `normalizeBetweenArrays` with `method="quantile"` from package limma.

Gquantile Calls `normalizeBetweenArrays` with `method="Gquantile"` from package limma.

Rquantile Calls `normalizeBetweenArrays` with `method="Rquantile"` from package limma.

Tquantile Calls `normalizeBetweenArrays` with `method="Tquantile"` from package limma.

Rquantile Calls `normalizeBetweenArrays` with `method="Rquantile"` from package limma.

vsn Calls `normalizeBetweenArrays` with `method="vsn"` from package limma.

loess Calls `normalize.loess` from package affy.

rankpercentile Calls `rankPercentile.normalize` from this package.

substract Calls `substract` from this package.

Author(s)

Benedikt Zacher <zacher@lmb.uni-muenchen.de>

See Also

`normalizeBetweenArrays`, `normalize.loess` , `substract`, `rankPercentile.normalize`

Examples

```
##
# dataPath <- system.file("extdata", package="Starr")
# bpmapChr1 <- readBpmap(file.path(dataPath, "Scerevisiae_tlg_chr1.bpmap"))

# cels <- c(file.path(dataPath,"Rpb3_IP_chr1.cel"), file.path(dataPath,"wt_IP_chr1.cel"),
#  file.path(dataPath,"Rpb3_IP2_chr1.cel"))
# names <- c("rpb3_1", "wt_1","rpb3_2")
# type <- c("IP", "CONTROL", "IP")
# rpb3Chr1 <- readCelFile(bpmapChr1, cels, names, type, featureData=TRUE, log.it=TRUE)

# rpb3_rankpercentile <- normalize.Probes(rpb3Chr1, method="rankpercentile")
```

plotBoxes *boxplots of experiments*

Description

Generates a boxplot of the of the given experiments.

Usage

```
plotBoxes(eSet, col=NULL)
```

Arguments

eSet	Either an ExpressionSet or a matrix, containing the data.
col	color, to fill the boxes

Author(s)

Benedikt Zacher <zacher@lmb.uni-muenchen.de>

See Also

boxplot

Examples

```
##
mat <- matrix(rnorm(1000000), ncol=2)
colnames(mat) <- c("Sample1", "Sample2")
mat[,1] <- mat[,1]-2
plotBoxes(mat)
```

plotcmarrt *Histogram of p-values and normal QQ plots for standardized MA statistics*

Description

Plot the histograms of p-values and normal QQ plots under correlation structure and independence.

Usage

```
plotcmarrt(cmarrt.ma, freq=FALSE)
```

Arguments

cmarrt.ma	output object from cmarrt.ma.
freq	see ?hist

Details

Diagnostic plots for comparing the distribution of standardized MA statistics under correlation and independence.

Value

Histogram of p-values and normal QQ plots under correlation structure and independence.

Note

If the normal quantile-quantile plot deviates from the reference line for unbound probes, this indicates that Gaussian approximation is not suitable for analyzing this data.

Author(s)

Pei Fen Kuan, Adam Hinz

References

P.F. Kuan, H. Chun, S. Keles (2008). CMARRT: A tool for the analysiz of ChIP-chip data from tiling arrays by incorporating the correlation structure. *Pacific Symposium of Biocomputing***13**:515-526.

See Also

cmarrt.ma,qqnorm

Examples

```
# dataPath <- system.file("extdata", package="Starr")
# bpmapChr1 <- readBpmap(file.path(dataPath, "Scerevisiae_tlg_chr1.bpmap"))

# cels <- c(file.path(dataPath,"Rpb3_IP_chr1.cel"), file.path(dataPath,"wt_IP_chr1.cel"),
#  file.path(dataPath,"Rpb3_IP2_chr1.cel"))
# names <- c("rpb3_1", "wt_1","rpb3_2")
# type <- c("IP", "CONTROL", "IP")
# rpb3Chr1 <- readCelFile(bpmapChr1, cels, names, type, featureData=TRUE, log.it=TRUE)

# ips <- rpb3Chr1$type == "IP"
# controls <- rpb3Chr1$type == "CONTROL"

# rpb3_rankpercentile <- normalize.Probes(rpb3Chr1, method="rankpercentile")
# description <- c("Rpb3vsWT")
# rpb3_rankpercentile_ratio <- getRatio(rpb3_rankpercentile, ips, controls, description, fkt=median, featureData=

# probeAnnoChr1 <- bpmapToProbeAnno(bpmapChr1)
```

```
# peaks <- cmarrt.ma(rpb3_rankpercentile_ratio, probeAnnoChr1, chr=NULL, M=NULL,250,window.opt=fixed.probe)

# plotcmarrt(peaks)
```

---

plotDensity *density plots of experiments*

Description

Generates a plot, showing the densities of the experiments.

Usage

```
plotDensity(eSet, oneDevice=T, main="")
```

Arguments

eSet	an ExprssionSet or a matrix, containing the data
oneDevice	should all lines be plotted to one device?
main	head of the plot

Author(s)

Benedikt Zacher <zacher@lmb.uni-muenchen.de>

See Also

plot.default, density

Examples

```
##
mat <- matrix(rnorm(1000000), ncol=2)
colnames(mat) <- c("Sample1", "Sample2")
mat[,1] <- mat[,1]-2
plotDensity(mat)
```

plotGCbias *Visualize GC-Bias of Hybridization*

Description

Generates a plot showing the GC-bias of the hybridization.

Usage

```
plotGCbias(intensity, sequence, main="")
```

Arguments

intensity	a vector of type numeric, containing the measured intensities.
sequence	a vector of type character, containing the sequences.
main	head of the plot

Author(s)

Benedikt Zacher <zacher@lmb.uni-muenchen.de>

See Also

boxplot

Examples

```
##
sequence <- unlist(lapply(1:50000, function(x) {paste(sample(c("A","T","C","G"),prob=c(0.3,0.3,0.2,0.2),25,repla
values <- runif(50000,min=-2,max=2)
plotGCbias(values, sequence)
```

plotImage *Reconstruct the array image*

Description

Function to visualize spatial distribution of raw intensities on Affymetrix Oligoarrays.

Usage

```
plotImage(cel)
```

Arguments

cel	a character, specifying the path to the CEL file

Author(s)

Benedikt Zacher <zacher@lmb.uni-muenchen.de>

See Also

readCel,levelplot

Examples

```
# dataPath <- system.file("extdata", package="Starr")
# plotImage(file.path(dataPath,"Rpb3_IP_chr1.cel"))
```

---

plotMA *M versus A plot*

Description

A matrix of M vs. A plots of each pair (ip, control) is produced.

Usage

```
plotMA(eSet, ip=NULL, control=NULL, col=NULL)
```

Arguments

eSet	an ExprssionSet or matrix, containing the data
ip	an integer, or boolean vector, that indicates, which columns in the ExpressionSet are IP experiments
control	an integer, or boolean vector, that indicates, which columns in the ExpressionSet are CONTROL or REFERENCE experiments
col	color, to fill the boxes

Author(s)

Benedikt Zacher <zacher@lmb.uni-muenchen.de>

See Also

ma.plot

Examples

```
##
mat <- matrix(rnorm(1000000), ncol=4)
colnames(mat) <- c("Sample1", "Sample2", "Sample3", "Sample4")
mat[,1] <- mat[,1]^2
plotMA(mat, c(TRUE, FALSE, TRUE, FALSE), c(FALSE, TRUE, FALSE, TRUE))
```

plotPosBias *Bias of hybridzation, depending on base position in sequence.*

Description

plotPosBias generates a plot showing the bias of hybridzation, depending on base position in sequence.

Usage

```
plotPosBias(intensity, sequence, main="", ylim)
```

Arguments

intensity	a vector of type numeric, containing the measured intensities
sequence	a vector of type character, containing the sequneces
main	head of the plot
ylim	ylim of plot

Author(s)

Benedikt Zacher <zacher@lmb.uni-muenchen.de>

Examples

```
##
sequence <- unlist(lapply(1:50000, function(x) {paste(sample(c("A","T","C","G"),prob=c(0.3,0.3,0.2,0.2),25,repla
values <- runif(50000,min=-2,max=2)
plotPosBias(values, sequence)
```

plotProfiles *Plotting ChIP profiles of one or more clusters*

Description

plotProfiles plots the ChIP profiles of one or more clusters. Additionally on can display the distribution of e.g. gene expression in the clusters.

Usage

```
plotProfiles(profiles, mfcol=NULL, mfrow=NULL, ylab="intensity", xlab="position", histograms=NULL, cl
```

Arguments

profiles	a list constructed by the function getProfiles().
mfcol	see ?par
mfrow	see ?par
ylab	see ?par
xlab	see ?par
histograms	a list of named vectors. Density plots are created for every vector and cluster.
cluster	A named integer vector, that maps the features to the cluster.
profileplot	should a clusterplot be shown?
meanprofile	should the mean profiles of each cluster be plotted??
...	arguments, passed to plot.default

Author(s)

Benedikt Zacher <zacher@lmb.uni-muenchen.de>

See Also

density, profileplot

Examples

```
##
sampls = 100
probes = 63
clus = matrix(rnorm(probes*sampls,sd=1),ncol=probes)
clus= rbind( t(t(clus)+sin(1:probes/10))+1:nrow(clus)/sampls , t(t(clus)+sin(pi/2+1:probes/10))+1:nrow(clus)/sam
clustering = kmeans(clus,3)$cluster
names(clustering) <- 1:length(clustering)


profiles <- apply(clus, 1, function(x) {list(upstream=x[1:20], region=x[21:43], downstream=x[44:63])})
names(profiles) <- 1:length(clustering)
profiles <- list(profile=profiles, upstream=20, downstream=20, borderNames=c("start", "stop"))

plotProfiles(profiles, cluster=clustering, ylim=c(-1,2.5), type="l", lwd=2)
```

---

plotRatioScatter *Plot ratios of all possible combinations of IP and CONTROL*

Description

A matrix of pairwise scatterplots of the ratios is created. The lower panel shows the correlation of the data.

Usage

```
plotRatioScatter(eSet, ip, control, density=F, sample=NULL, cluster=T, cex=1)
```

Arguments

eSet	an ExprssionSet or matrix, containing the data
ip	an integer, or boolean vector, that indicates, which columns in the ExpressionSet are IP experiments
control	an integer, or boolean vector, that indicates, which columns in the ExpressionSet are CONTROL or REFERENCE experiments
density	if TRUE, a density scatter plot is plotted. This plot shows the density of the data.
sample	An integer, indicating the number of subsamples to take for the density scatterplot. This is only recommended if the data is very large, as the density computation takes some time. #
cluster	if cluster=T, the experiments are clustered and similiar experiments are plotted together.
cex	see ?par

Author(s)

Benedikt Zacher <zacher@lmb.uni-muenchen.de>

See Also

pairs, densityscatter

Examples

```
##
points <- 10^4
x <- rnorm(points/2)
x <- c(x,x+2.5)
x <- sign(x)*abs(x)^1.3
y <- x + rnorm(points,sd=0.8)
z <- y*2
mat <- matrix(c(x,y,z), ncol=3)
colnames(mat) <- c("A", "B1", "B2")
plotRatioScatter(mat, c(TRUE, FALSE, FALSE), c(FALSE, TRUE, TRUE), density=TRUE)
```

plotScatter *High level scatterplot of experiments*

Description

A matrix of pairwise scatterplots is created. The lower panle shows the correlation of the data.

Usage

```
plotScatter(eSet, density=F, cluster=T, sample=NULL, cex=1)
```

Arguments

eSet	an ExprssionSet or matrix, containing the data
density	if TRUE, a density scatter plot is plotted. This plot shows the density of the data.
sample	An integer, indicating the number of subsamples to take for the density scatterplot. This is only recommended if the data is very large, as the density computation takes some time.
cluster	if cluster=T, the experiments are clustered and similiar experiments are plotted together.
cex	see ?par

Author(s)

Benedikt Zacher <zacher@lmb.uni-muenchen.de>

See Also

pairs, densityscatter

Examples

```
##
points <- 10^4
x <- rnorm(points/2)
x <- c(x,x+2.5)
x <- sign(x)*abs(x)^1.3
y <- x + rnorm(points,sd=0.8)
mat <- matrix(c(x,y), ncol=2)
colnames(mat) <- c("a", "b")
plotScatter(mat, density=TRUE)
```

profileplot *Vizualize clusters*

Description

Visualization of a set of "profiles" (i.e. a consecutive series of measurements like a time series, or the DNA binding levels along different positions on a gene). The profiles are given as the rows of a (samples x positions) matrix that contains the measurements. Instead of plotting a line for each profile (row of the matrix), the q-quantiles for each position (column of the matrix) are calculated, where q runs through a set of representative quantiles. Then for each q, a line of q-quantiles is plotted along the positions. Color coding of the quantile profiles aids the interpretation of the plot: There is a color gradient from the median profile to the 0 (=min) resp. 1(=max) quantile.

Usage

```
profileplot(cluster, label=NULL,at=NULL, main = "", xlim=NULL, xlab = "", xaxt = "s",xlabels = NULL, las
```

Arguments

cluster	a (samples x columns) matrix with numerical entries. Each sample row is understood as a consecutive series of measurements. Missing values are not allowed so far
label	if multiple clusters should be plotted in one diagram, the cluster labels for each item are given in this vector
at	optional vector of length ncol(cluster), default = 1:ncol(cluster). Specifies the x-values at which the positions will be plotted.
main	the title of the plot, standard graphics parameter
xlim	xlimits, standard graphics parameter
xlab	x-axis legend, standard graphics parameter
xaxt	should an x axis be plotted at all? (="n" if not), standard graphics parameter
xlabels	character vector. If specified, this text will be added at the "at"-positions as x-axis labels.
las	direction of the xlabels text. las=1: horizontal text, las=2: vertical text
ylim	ylimits, standard graphics parameter
ylab	y-axis legend, standard graphics parameter
fromto	determines the smallest and the largest quantile that are plotted in colors, more distant values are plotted as outliers
colpal	either "red","green","blue" (predefined standard color palettes in profileplot), or a vector of colors to be used instead.
nrcolors	not very important. How many colors will the color palette contain? Usually, the default = 25 is sufficient
outer.col	color of the outlier lines, default = "light grey". For no outliers, choose outer.col="none"
add.quartiles	should the quartile lines be plotted (grey/black)? default=TRUE
add	should the profile plot be added to the current plot? Defaults to FALSE
separate	should each cluster, be plotted in a separate window? Defaults to TRUE

Author(s)

Achim Tresch, Benedikt Zacher <tresch@lmb.uni-muenchen.de>

Examples

```
sampls = 100
probes = 63
at = (-31:31)*14
clus = matrix(rnorm(probes*sampls,sd=1),ncol=probes)
clus= rbind( t(t(clus)+sin(1:probes/10))+1:nrow(clus)/sampls , t(t(clus)+sin(pi/2+1:probes/10))+1:nrow(clus)/sam
labs = paste("cluster",kmeans(clus,4)$cluster)

profileplot(clus,main="All data",fromto=c(0,1))
profileplot(clus,label=labs,main="Clustered data",colpal=c("heat","blue","red","topo"),add.quartiles=FALSE)
profileplot(clus,main="Same data, 4 clusters in one plot\n color gradient fromto = c(0.4,0.6), no outliers plotted"
colpal=c("heat","blue","red","green"),outer.col="none")
```

---

read.gffAnno *Reading gff annotation*

Description

This functions reads the annotation from a gff file.

Usage

```
read.gffAnno(gffFile, feature=NULL)
```

Arguments

gffFile	path to file
feature	feature to select ("character"). If feature="gene", then only rows, representing this feature are read.

Author(s)

Benedikt Zacher <zacher@lmb.uni-muenchen.de>

Examples

```
##
# dataPath <- system.file("extdata", package="Starr")
# transcriptAnno <- read.gffAnno(file.path(dataPath, "transcriptAnno.gff"), feature="transcript")
```

`readCelFile` *Read raw intensities from CEL files*

Description

Function to read the raw intensities of the perfect match probes (PM) of Affymetrix CEL files into an ExpressionSet. This function is used to read one-color data. For two-color data use the functions from the Ringo package.

Usage

```
readCelFile(bpmap, cel_files, names, type, experimentData=NULL, featureData=T, log.it=T, phenodata=NUL
```

Arguments

`bpmap`	Either a list, created by the function readBpmap() from the affy package, or the path to the bpmap file.
`cel_files`	a character vector, specifying the path to the CEL files
`names`	a character vector, containing the names of the experiments
`type`	a character vector, containing the type of experiment, e.g. "IP" for an Immunoprecipitation, or "CONTROL" for a control or reference experiment was done
`experimentData`	This must be an object of type MIAME, which details information about e.g., the investigator or lab where the experiment was done, an overall title, and other notes
`featureData`	If TRUE, a featureData object is added to the ExpressionSet, containing information about the chromosome, position in the genome and sequence of the features
`log.it`	If TRUE, logged intesities are read
`phenodata`	data.frame, containing columns name, type, CEL.

Value

Returns raw intensity values in form of an ExpressionSet with additional information:

`assayData`	This object contains the measured probe intensities.
`phenoData`	contains further description of the experiments, such as names or type
`featureData`	containing information about the chromosome, position in the genome and sequence of the features
`experimentData`	details information about e.g., the investigator or lab where the experiment was done

Author(s)

Benedikt Zacher <zacher@lmb.uni-muenchen.de>

See Also

readCelIntensities, xy2indices

Examples

```
##
# dataPath <- system.file("extdata", package="Starr")
# bpmapChr1 <- readBpmap(file.path(dataPath, "Scerevisiae_tlg_chr1.bpmap"))

# cels <- c(file.path(dataPath,"Rpb3_IP_chr1.cel"), file.path(dataPath,"wt_IP_chr1.cel"),
#  file.path(dataPath,"Rpb3_IP2_chr1.cel"))
# names <- c("rpb3_1", "wt_1","rpb3_2")
# type <- c("IP", "CONTROL", "IP")
# rpb3Chr1 <- readCelFile(bpmapChr1, cels, names, type, featureData=TRUE, log.it=TRUE)
```

---

remap *Remap reporter sequences to the genome and create a new bpmap file*

Description

This function remaps the reporter sequences on the chip on the genome and outputs a new bpmap annotation, containing only unique matches to the genome. A remapping is recommended if the bpmap file was built on an outdated genome, or if sequences, that match the genome more than once should be excluded.

Usage

```
remap(bpmap=NULL, seqs=NULL, nseq=NULL, path="", complementary=FALSE, reverse=FALSE, reverse_compleme
```

Arguments

bpmap	A list, created by the function readBpmap() from the affy package.
nseq	Number of sequences, that are searched in one iteration.
seqs	Sequences to search as a character vector
path	path to genomic fasta files
complementary	If TRUE, the sequences are searched in the complementary strand of the text
reverse	If TRUE, the sequences are searched in the reverse strand of the text
reverse_complementary	If TRUE, the sequences are searched in the reverse complementary strand of the text
return_bpmap	If TRUE, the output is a list in bpmap format

Author(s)

Benedikt Zacher <zacher@lmb.uni-muenchen.de>

Examples

```
# dataPath <- system.file("extdata", package="Starr")

# bpmapChr1 <- readBpmap(file.path(dataPath, "Scerevisiae_tlg_chr1.bpmap"))
# newbpmap <- remap(bpmapChr1, nseq=5000000, path=dataPath, reverse_complementary=TRUE, return_bpmap=TRUE)
```

`writeGFF` *write ChIP-chip data to a gff file*

Description

This function writes the all columns of the assayData to a gff file.

Usage

```
writeGFF(expressionSet, probeAnno, file)
```

Arguments

`expressionSet`	an ExpressionSet object
`probeAnno`	a probeAnno object
`file`	path to write to

Author(s)

Benedikt Zacher <zacher@lmb.uni-muenchen.de>

`writePosFile` *Creating a pos file*

Description

Writes a Nimblegen pos file from a given Affymetrix bpmap file.

Usage

```
writePosFile(bpmap, file)
```

Arguments

`bpmap`	Either a list, created by the function readBpmap() from the affy package. Or a path to the bpmap file.
`file`	a character, specifying the path to the file to be written

Author(s)

Benedikt Zacher <zacher@lmb.uni-muenchen.de>

writeWIG *write ChIP-chip data to a *.wig file*

Description

This function writes the all columns of the assayData to a wiggle file.

Usage

```
writeWIG(expressionSet, probeAnno, file, chr=NULL, probeLength=NULL)
```

Arguments

expressionSet	an ExpressionSet object
probeAnno	a probeAnno object
file	path to write to
chr	subset of chromosomes in probeAnno. If specified, only the subset is written to the file.
probeLength	length of the probes on the chip.

Author(s)

Benedikt Zacher <zacher@lmb.uni-muenchen.de>

Index