charm

April 20, 2011

`bgAdjust` *Remove background*

Description

Estimate and remove background signal using anti-genomic background probes

Usage

```
bgAdjust(dat, copy=TRUE)
```

Arguments

`dat`	a TilingFeatureSet
`copy`	Only relevant when using disk-backed objects. If TRUE a copy will be made leaving the original object (dat) unchanged. The input object will not be preserved if copy=FALSE

Details

Background signal removal using a modified version of the RMA convolution model. The background signal level is estimated within GC-strata using anti-genomic background probes.

Value

a TilingFeatureSet

Author(s)

Martin Aryee <aryee@jhu.edu>

Examples

```
# See normalizeBetweenSamples
```

countGC *Count probe GC content*

Description

Return the GC content for each probe

Usage

```
countGC(dat, type = "pm", idx)
```

Arguments

dat	a TilingFeatureSet object
type	pm or bg probes
idx	An optional vector of probe indices for which to return GC content. If not specified, values for all pm (or bg) probes will be returned.

Details

This function returns the sum of #G + #C in the pm or bg probes.

Value

a numeric vector

Author(s)

Martin Aryee <aryee@jhu.edu>

See Also

readCharm

Examples

```
if (require(charmData)) {
phenodataDir <- system.file("extdata", package="charmData")
pd <- read.delim(file.path(phenodataDir, "phenodata.txt"))
pd <- subset(pd, sampleID=="441_liver")
dataDir <- system.file("data", package="charmData")
setwd(dataDir)
rawData <- readCharm(files=pd$filename, sampleKey=pd)
ngc <- countGC(rawData)
head(ngc)
}
```

`cpgdensity` *Get CpG density for genomic regions*

Description

Calculate the CpG density for a set of windows

Usage

```
cpgdensity(subject, chr, pos, windowSize = 500, sequence = "CG")
```

Arguments

`subject`	BSGenome object (e.g. Hsapiens)
`chr`	character vector
`pos`	numeric vector
`windowSize`	number value
`sequence`	character string

Details

Calculate the CpG density for a set of regions. chr and pos specify the region mid-points and windowSize specifies the size of the window to be centered on these mid-points. i.e. The window will stretch from pos-windowSize/2 to pos+windowSize/2.

Value

a numeric vector

Author(s)

Martin Aryee <aryee@jhu.edu>

Examples

```
if (require(BSgenome.Hsapiens.UCSC.hg18)){
chr <- c("chr1", "chr1", "chr2")
pos <- c(100000, 100500, 100000)
cpgd <- cpgdensity(Hsapiens, chr=chr, pos=pos, windowSize = 500)
cpgd
}
```

dmrFdr *Calculate FDR q-values for differentially methylated regions (DMRs)*

Description

Estimate false discovery rate q-values for a set of differentially methylated regions using a permutation approach.

Usage

```
dmrFdr(dmr, compare = 1, numPerms = 1000, seed = NULL, verbose = TRUE)
```

Arguments

dmr	a dmr object as returned by dmrFinder
compare	The dmr table for which to calculate DMRs. See details.
numPerms	Number of permutations
seed	Random seed (for reproducibility)
verbose	Boolean

Details

This function estimates false discovery rate q-values for a dmr object returned by dmrFinder. dmrFinder can return a set of DMR tables with one or more pair-wise comparisons between groups. dmrFdr currently only calculated q-values for one of these at a time. The dmr table to use (if the dmr object contains more than one) is specified by the compare option.

Value

a list object in the same format as the input, but with extra p-val and q-val columns for the tabs element.

Author(s)

Martin Aryee <aryee@jhu.edu>

See Also

dmrFinder, dmrPlot, regionPlot

Examples

```
if (require(charmData) & require(BSgenome.Hsapiens.UCSC.hg18)) {
phenodataDir <- system.file("extdata", package="charmData")
pd <- read.delim(file.path(phenodataDir, "phenodata.txt"))
pd <- subset(pd, tissue %in% c("liver", "colon"))
# Validate format of sample description file
res <- validatePd(pd)
dataDir <- system.file("data", package="charmData")
setwd(dataDir)
# Read in raw data
rawData <- readCharm(files=pd$filename, sampleKey=pd)
```

```
# Find non-CpG control probes
ctrlIdx <- getControlIndex(rawData, subject=Hsapiens)
# Estimate methylation
p <- methp(rawData, controlIndex=ctrlIdx)
# Find differentially methylated regions
grp <- pData(rawData)$tissue
dmr <- dmrFinder(rawData, p=p, groups=grp,
compare=c("liver", "colon"), cutoff=0.95)
head(dmr$tabs[[1]])
# Estimate false discovery rate for DMRs
dmr <- dmrFdr(dmr, numPerms=3, seed=123)
head(dmr$tabs[[1]])

                ##Not run:
                ## Plot top 10 DMRs:
                #dmrPlot(dmr=dmr, which.table=1, which.plot=1:10, legend.size=1, all.line
                ## plot any given genomic regions using this data, supplying the regions
                #mytab = data.frame(chr=as.character(c(dmr$tabs[[1]]$chr[1],"chrY",dmr$ta
                #regionPlot(tab=mytab, dmr=dmr, outfile="./myregions.pdf", which.plot=1:5
                ## note that region 2 is not plotted since it is not on the array.
}
```

`dmrFinder` *Find differentially methylated regions (DMRs)*

Description

Find differentially methylated regions (DMRs) from tiling microarray data.

Usage

```
dmrFinder(eset=NULL, groups, p=NULL, l=NULL, chr=NULL, pos=NULL, pns=NULL,
sdBins=NULL, controlIndex=NULL,
controlProbes=c("CONTROL_PROBES", "CONTROL_REGIONS"), Indexes=NULL,
filter=NULL, package=NULL, ws=7, verbose=TRUE, compare="all",
withinSampleNorm="loess", betweenSampleNorm="quantile",
cutoff=0.995, sortBy="ttarea",paired=FALSE,pairs=NULL,DD=NULL,COMPS=NULL,
        removeIf=expression(nprobes<2),...)
```

Arguments

`eset`	a TilingFeatureSet
`groups`	a vector of group labels for the samples in eset
`p`	a matrix of percentage methylation values (scale: 0, 1). One column per sample
`l`	a matrix of methylation values (scale: -Inf, Inf), typically log-ratios.
`chr`	vector of chromosome labels for the probes in eset, p or l
`pos`	vector of chromosomal coordinates for the probes in eset, p or l
`pns`	vector of region names for the probes in eset, p or l
`sdBins`	not currently implemented
`controlIndex`	vector of indices of non-CpG control probes

`controlProbes`
: not currently used

`Indexes`
: not currently used

`filter`
: smoothing window weights. See details

`package`
: annotation package name

`ws`
: smoothing window size parameter. See details.

`verbose`
: Verbose progress reporting

`compare`
: the groups between which to find DMRs.

`withinSampleNorm`
: within-sample normalization method. "loess" or "none"

`betweenSampleNorm`
: between-sample normalization method. "quantile", "sqn" or "none"

`cutoff`
: t-statistic cutoff used to identify probes as being in a DMR

`sortBy`
: sort column for the DMR table. "area", "ttarea", "avg.diff", or "max.diff".

`paired`
: if TRUE, do comparisons within pairs of samples. FALSE by default.

`pairs`
: if paired=TRUE, this must be provided. a vector of pair identifiers for the samples in eset. values must be the same within pairs and different between pairs.

`DD`
: DD object returned by dmrFinder when paired=TRUE. This argument may be ignored.

`COMPS`
: comps object returned by dmrFinder. This argument may be ignored.

`removeIf`
: expression indicating which DMRs to drop from the DMR tables that get returned. The negation of this is used as the subset argument to the subset function when it is called on the final DMR table before it is returned. If NULL, no DMRs will be subsetted out from the final table before it is returned. DMR table column names to use are listed below. E.g., to drop all DMRs with less than 4 probes, set removeIf=expression(nprobes<4).

`...`
: further options to be passed to methp

Details

This function finds differentially methylated regions (DMRs). The sortBy parameter can be used to sort the DMRs by area (# probes x average difference), t-statistic area (# probes x average t-statistic), average difference, or maximum difference.

Value

A list with

`tabs`
: A list of DMR tables, one per comparison with columns:

chr chromosome of DMR (bp)

start start of DMR (bp)

end end of DMR (bp)

p1 if paired=FALSE, and p!=NULL or l=NULL, average percentage methylation of all probes between start and end for group 1

p2 if paired=FALSE, and p!=NULL or l=NULL, average percentage methylation of all probes between start and end for group 2

m1 if paired=FALSE, p=NULL and l!=NULL, average methylation l (logit(percentage methylation) if l=NULL) of all probes between start and end for group 1

m2 if paired=FALSE, p=NULL and l!=NULL, average methylation l (logit(percentage methylation) if l=NULL) of all probes between start and end for group 2

regionName name of the tiling region in which the DMR is found (These names come from the NDF file)

indexStart index of first probe in DMR

indexEnd index of last probe in DMR

nprobes number of probes for the DMR, i.e., indexEnd-indexStart+1

diff average percentage methylation difference within the DMR if paired=FALSE, and average l (logit(percentage) methylation if l=NULL) difference within the DMR if paired=TRUE

maxdiff maximum percentage methylation difference within the DMR if paired=FALSE, and maximum l (logit(percentage) methylation if l=NULL) difference within the DMR if paired=FALSE

area nprobes x average difference

ttarea nprobes x (average probe level t-statistic for between group difference)

`p` A matrix of percentage methylation estimates (NOTE: the probe order may differ from that of the input p matrix since probes are sorted into chromosomal order)

`l` This contains methylation log-ratios if they were passed to the function. Otherwise it contains logit-transformed percentage methylation estimates. (NOTE: the probe order may differ from that of the input l matrix since probes are sorted into chromosomal order)

`chr` a vector of chromosomes corresponding to the rows of p and l

`pos` a vector of positions corresponding to the rows of p and l

`pns` a vector of probe region names corresponding to the rows of p and l

`controlIndex` a vector of control probe indices

`gm` if paired=FALSE, group medians of the l matrix

`DD` if paired=TRUE, a list of within-pair differences for each comparison

`sMD` if paired=TRUE, a matrix of smoothed mean within-pair differences for each comparison

`groups` a vector of group labels

`args` the DMR finder parameter vector

`comps` the vector of pairwise group comparisons

`package` the array annotation package name

Author(s)

Martin Aryee <aryee@jhu.edu>, Peter Murakami, Rafael Irizarry

See Also

`readCharm`, `methp`, `dmrFdr`

Examples

```
# See dmrFdr
```

`dmrPlot` *Plot differentially methylated regions (DMRs)*

Description

Plot differentially methylated regions (DMRs) from tiling microarray data.

Usage

```
dmrPlot(dmr, which.table=1:length(dmr$tabs), which.plot=1:30, legend.size=1, all
```

Arguments

`dmr`	a list object as returned by dmrFinder.
`which.table`	a vector of indices identifying which tables in the dmr list to plot regions from.
`which.plot`	a vector of indices identifying which regions (rows) from each table to plot.
`legend.size`	cex argument for the legend (factor by which to magnify/shrink the legend).
`all.lines`	if TRUE, plot the smooth lines for all groups. If FALSE, only for the 2 groups being compared.
`all.points`	if TRUE, plot the points for all groups. If FALSE, only for the 2 groups being compared.
`colors.l`	a vector of line colors, one color for each group whose line is to be plotted (in alphabetical order).
`colors.p`	a vector of point colors, one color for each group whose points are to be plotted (in alphabetical order).
`outpath`	where to save the output pdf file.
`plot.p`	set to FALSE if you want to plot the methlation values (the "l" output from dmrFinder) instead of the percentage methylation values (the "p" output). If dmrFinder was run on l instead of p, plot.p=FALSE necessarily.

Details

This function plots the differentially methylated regions (DMRs).

Author(s)

Martin Aryee <aryee@jhu.edu>, Peter Murakami, Rafael Irizarry

See Also

`regionPlot`, `dmrFinder`, `dmrFdr`

Examples

```
# See dmrFdr
```

`getControlIndex` *Get indices of control probes from CpG-free regions*

Description

Get indices of control probes from CpG-free regions.

Usage

```
getControlIndex(dat, controlProbes = c("CONTROL_PROBES", "CONTROL_REGIONS"),
noCpGWindow = 1000, subject, onlyGood = FALSE, matrix = TRUE)
```

Arguments

`dat`	TilingFeatureSet
`controlProbes`	vector of names used to denote control probes in the 'container' column of the Nimblegen annotation (ndf) file. Optional
`noCpGWindow`	Size of the window centered on the probe that must be CpG-free
`subject`	A BSgenome object
`onlyGood`	deprecated option
`matrix`	deprecated option

Details

The probes can either be identified as control probes in the microarray annotation package, or alternatively the function will search the genome (given an appropriate BSgenome object) for suitable probes.

Value

a vector

Author(s)

Martin Aryee <aryee@jhu.edu>

Examples

```
# See dmrFdr
```

`methPercent` *Estimate percentage DNA methylation from log-ratios*

Description

Estimate percentage DNA methylation from log-ratios

Usage

```
methPercent(m, pmIndex, ngc, commonParams = TRUE)
```

Arguments

`m`	a matrix of M-values (methylation log-ratios). One column per sample.
`pmIndex`	A vector of probe indices to use in the calculation. Usually set to the indices of the pm probes (excluding background and other non-specific controls) by using pmIndex=pmindex(dat)
`ngc`	a vector with GC-content of probes. Same length as nrow(m)
`commonParams`	boolean indicating whether a common set of parameters should be used for all samples when converting M-values to percentage methylation.

Details

This function estimates percentage DNA methylation from normalized methylation log-ratios (M-values).

Value

a matrix of percentage methylation estimates. Same dimensions as m

Author(s)

Martin Aryee <aryee@jhu.edu>

Examples

```
if (require(charmData) & require(BSgenome.Hsapiens.UCSC.hg18)) {
phenodataDir <- system.file("extdata", package="charmData")
pd <- read.delim(file.path(phenodataDir, "phenodata.txt"))
pd <- subset(pd, sampleID=="441_liver")
dataDir <- system.file("data", package="charmData")
setwd(dataDir)
# Read in raw data
rawData <- readCharm(files=pd$filename, sampleKey=pd)
# Find non-CpG control probes
ctrlIdx <- getControlIndex(rawData, subject=Hsapiens)
# Get normalized methylation log-ratios
m <- methp(rawData, controlIndex=ctrlIdx, returnM=TRUE)
# Estimate percentage methylation
ngc <- countGC(rawData)
p <- methPercent(m, ngc=ngc)
}
```

methp *Estimate DNA methylation*

Description

Estimate DNA methylation from McrBC/CHARM microarray data in terms of log-ratios or percentages.

Usage

```
methp(dat, spatial = TRUE, bgSubtract = TRUE, withinSampleNorm = "loess",
scale = c(0.99, 0.99), betweenSampleNorm = "quantile",
controlProbes = c("CONTROL_PROBES", "CONTROL_REGIONS"),
controlIndex = NULL, excludeIndex = NULL,
commonMethPercentParams = NULL,
verbose = TRUE, returnM = FALSE,
plotDensity = NULL, plotDensityGroups = NULL)
```

Arguments

dat	a TilingFeatureSet object
spatial	boolean indicating whether to correct spatial artefacts
bgSubtract	boolean indicating whether to estimate and remove background signal before computing log-ratios
withinSampleNorm	within-sample normalization method. Choices are "loess" and "none"
scale	a numeric vector (x,y). The xth percentile of each sample is scaled to represent y% methylation. The default c(0.99, 0.99) means probes in the 99% percentile represent 99% methylation.
betweenSampleNorm	between-sample normalization method. Choices are "quantile", "sqn", and "none". See Details for more fine-grained control.
controlProbes	character string of the label assigned to non-CpG control probes in the annotation file (i.e. the container column of the .ndf file).
controlIndex	a vector of non-CpG control probe indices
excludeIndex	a vector of probe indices indicating which pm probes to ignore when creating normalization target distributions.
commonMethPercentParams	boolean indicating whether a common set of parameters should be used for all samples when converting M-values to percentage methylation.
verbose	boolean: Verbose output?
returnM	boolean. Return M-values without converting to percentage methylation estimates
plotDensity	if specified this is the filename of the pdf diagnostic density plots.
plotDensityGroups	numeric vector of group labels used to color lines in the diagnostic density plots (see plotDensity option)

Details

This function provides probe-level estimates of percentage DNA methylation from CHARM microarray data.

Value

A matrix of probe-level percentage methylation estimates, one column per sample.

Author(s)

Martin Aryee <aryee@jhu.edu>

See Also

readCharm

Examples

```
# See dmrFdr
```

---

normalizeBetweenSamples

Between-sample normalization

Description

Between-sample normalization for two-color DNA methylation microarray data.

Usage

```
normalizeBetweenSamples (dat, copy=TRUE,
m="allQuantiles", untreated="none", enriched="none",
controlProbes=c("CONTROL_PROBES", "CONTROL_REGIONS"),
controlIndex=NULL, excludeIndex=NULL, verbose=FALSE)
```

Arguments

dat	a TilingFeatureSet object
copy	Only relevant when using disk-backed objects. If TRUE a copy will be made leaving the original object (dat) unchanged. The input object will not be preserved if copy=FALSE
m	normalization method for log-ratios. "allQuantiles" for full quantile normalization, or "none"
untreated	normalization method for the untreated channel. "complete", "allQuantiles" or "none"
enriched	normalization method for the untreated channel. "sqn", "allQuantiles" or "none"
controlProbes	character string of the label assigned to non-CpG control probes in the annotation file (i.e. the container column of the .ndf file).
controlIndex	a vector of non-CpG control probe indices

excludeIndex a vector indicating which pm probes to ignore when creating normalization target distributions. Can be a vector of probe indices or a boolean vector of length(pmindex(dat)).

verbose boolean: Verbose output?

Details

This function is used by methp performs between-sample normalization. It is normally not used directly by the user.

Value

a TilingFeatureSet

Author(s)

Martin Aryee <aryee@jhu.edu>

See Also

methp

Examples

```
if (require(charmData) & require(BSgenome.Hsapiens.UCSC.hg18)) {
phenodataDir <- system.file("extdata", package="charmData")
pd <- read.delim(file.path(phenodataDir, "phenodata.txt"))
pd <- subset(pd, sampleID=="441_liver")
dataDir <- system.file("data", package="charmData")
setwd(dataDir)
rawData <- readCharm(files=pd$filename, sampleKey=pd)
# Correct spatial artifacts
dat <- spatialAdjust(rawData)
# Remove background signal
dat <- bgAdjust(dat)
# Find non-CpG control probes
ctrlIdx <- getControlIndex(rawData, subject=Hsapiens)
# Within-sample normalization
dat <- normalizeWithinSamples(dat, controlIndex=ctrlIdx)
# Within-sample normalization
dat <- normalizeBetweenSamples(dat)
}
```

---

normalizeWithinSamples

Within-sample normalization for two-color data

Description

Within-sample (between-channel) normalization for two-color DNA methylation microarray data.

Usage

```
normalizeWithinSamples(dat, copy=TRUE,
method = "loess", scale=c(0.99, 0.99),
controlProbes = c("CONTROL_PROBES", "CONTROL_REGIONS"),
controlIndex = NULL, approx=TRUE, breaks=1000, verbose=FALSE)
```

Arguments

dat	a TilingFeatureSet
copy	Only relevant when using disk-backed objects. If TRUE a copy will be made leaving the original object (dat) unchanged. The input object will not be preserved if copy=FALSE
method	normalization method. "loess" or "none"
scale	a numeric vector (x,y). The xth percentile of each sample is scaled to represent y% methylation. The default c(0.99, 0.99) means probes in the 99% percentile represent 99% methylation. Set to NA for no scaling.
controlProbes	character string of the label assigned to non-CpG control probes in the annotation file (i.e. the container column of the .ndf file).
controlIndex	a vector of non-CpG control probe indices
approx	Bin probes by signal intensity when loess normalizing. Much faster when TRUE
breaks	Number of bins to use when approx=TRUE
verbose	boolean: Verbose output?

Details

This function is used by methp performs within-sample (between-channel) normalization. It is normally not used directly by the user.

Value

a TilingFeatureSet

Author(s)

Martin Aryee <aryee@jhu.edu>, Rafael Irizarry

Examples

```
# See normalizeBetweenSamples
```

`plotDensity` *Log-ratio density plot for all probes and control probes*

Description

Make density plots of log-ratios for two-color microarray data. Two plots are produced: one for all probes on the array, and a second for the control probes.

Usage

```
plotDensity(dat, rx = c(-4, 6), controlIndex = NULL, controlProbes=NULL,
            pdfFile = NULL, main = NULL, lab=NULL)
```

Arguments

`dat`	a TilingFeatureSet
`rx`	x-axis range
`controlIndex`	a vector of non-CpG control probe indices
`controlProbes`	vector of names used to denote control probes in the 'container' column of the Nimblegen annotation (ndf) file.
`pdfFile`	name of output pdf file
`main`	main title
`lab`	vector of sample labels. If not specified the sample names from dat will be used.

Details

This function makes density plots for a) all probes and b) control probes. It is typically called from within methp when a file name is specified for its plotDensity option. The plots are useful for identifying problematic outlier samples.

Value

No return value. Called for its side-effect of producing a pdf plot.

Author(s)

Martin Aryee <aryee@jhu.edu>

Examples

```
if (require(charmData) & require(BSgenome.Hsapiens.UCSC.hg18)) {
phenodataDir <- system.file("extdata", package="charmData")
pd <- read.delim(file.path(phenodataDir, "phenodata.txt"))
# Read in raw data
dataDir <- system.file("data", package="charmData")
rawData <- readCharm(path=dataDir, files=pd$filename,
 sampleKey=pd)
ctrlIdx <- getControlIndex(rawData, subject=Hsapiens)
plotDensity(rawData, controlIndex=ctrlIdx, pdfFile="density.pdf")
}
```

qcReport *Microarray quality report*

Description

Calculate microarray quality scores and produce an optional pdf report

Usage

```
qcReport(dat, file = NULL, utRange = c(30, 100), enRange = c(8, 12),
numProbes = 5e+05, blockSize)
```

Arguments

dat	a TilingFeatureSet
file	name of output pdf file
utRange	color-scale range for the untreated channel plots
enRange	color-scale range for the methyl-depleted channel plots
numProbes	maximum number of probes to use for plots. If smaller than the number of probes on the array numProbes are chosen at random, speeding up calculations for high-density arrays with several million probes.
blockSize	The array is divided into a series of blockSize x blockSize rectangular blocks and the average signal level calculated for each. If blockSize is unspecified a size is chosen that gives about 1250 probes per block.

Details

This function calculates microarray quality scores and produces an optional pdf report. Three quality metrics are calculated for each array:

Average signal strength. The average percentile rank of untreated channel signal probes among the background (anti-genomic) probes. Since the untreated channel contains total DNA a successful hybridization would have strong signal for all untreated channel genomic probes.

Untreated channel signal standard deviation. The array is divided into a series of rectangular blocks and the average signal level calculated for each. Since probes are arranged randomly on the array there should be no large differences between blocks. Arrays with spatial artifacts have a larger standard deviation between blocks.

Methyl-depleted channel signal standard deviation

Value

a matrix with a row for each sample. The 3 columns contain array signal strength score, untreated channel standard deviation and methyl-depleted channel standard deviation.

Author(s)

Martin Aryee <aryee@jhu.edu>

Examples

```
if (require(charmData)) {
phenodataDir <- system.file("extdata", package="charmData")
pd <- read.delim(file.path(phenodataDir, "phenodata.txt"))
dataDir <- system.file("data", package="charmData")
setwd(dataDir)
rawData <- readCharm(files=pd$filename, sampleKey=pd)
qcReport(rawData, file="qcReport.pdf")
}
```

---

readCharm *Read in McrBC/CHARM DNA methylation microarray data*

Description

Read in DNA methylation microarray data from the McrBC/CHARM platform

Usage

```
readCharm(files, path = ".", ut = "_532.xys", md = "_635.xys",
sampleKey, sampleNames = NULL, pkgname, type = NULL, ...)
```

Arguments

files	a vector of xys filenames
path	the path to the xys files
ut	the file ending that designates untreated channel files
md	the file ending that designates methyl-depleted channel files
sampleKey	a data frame with sample description information. One line per xys file.
sampleNames	a vector of names to use for the samples. One line per xys file.
pkgname	the annotation package name
type	deprecated option
...	additional options passed on to read.xysfiles2

Details

This function is a convenience wrapper to read.xysfiles2 to simplify reading in DNA methylation data from the Nimblegen McrBC/CHARM microarray platform. It makes guesses about the extensions used for the methyl-depleted (md) and untreated channels (ut).

Value

A TilingFeatureSet object.

Author(s)

Martin Aryee <aryee@jhu.edu>

References

www.biostat.jhsph.edu/~maryee/charm

See Also

methp, dmrFinder

Examples

```
# See normalizeBetweenSamples
```

`regionPlot` *Plot user-supplied genomic regions.*

Description

Plot any given genomic regions from tiling microarray data.

Usage

```
regionPlot(tab, dmr, outfile, which.plot, which.groups=colnames(dmr$gm), cl=2:(n
```

Arguments

`tab`	a data frame with columns chr, start, and end identifying the regions to be plotted from the data.
`dmr`	a list object as returned by dmrFinder, providing the data to be plotted.
`outfile`	a character string giving the name of the pdf file that will be saved. Include the full path if file is not to be saved in the current working directory.
`which.plot`	a vector of indices identifying which regions (rows) from tab to plot.
`which.groups`	a character vector of names (or a numeric vector of indices for the columns of dmr$gm) identifying which groups to plot.
`cl`	a vector of line and point colors, one for each group in which.groups in alphabetical order by group name.
`legend.size`	cex argument for the legend (factor by which to magnify/shrink the legend).
`buffer`	An integer to control how many basepairs to show on either side of the plotted regions.
`plot.p`	set to FALSE if you want to plot the methlation values (the "l" output from dmrFinder) instead of the percentage methylation values (the "p" output). If dmrFinder was run on l instead of p, plot.p=FALSE necessarily.

Details

This function enables plotting of any regions, not just DMRs.

Author(s)

Martin Aryee <aryee@jhu.edu>, Peter Murakami, Rafael Irizarry

See Also

dmrPlot, dmrFinder, dmrFdr

Examples

```
# See dmrFdr
```

`spatialAdjust` *Correct spatial artifacts*

Description

Remove spatial artifacts from microarray data stored in TilingFeatureSet objects

Usage

```
spatialAdjust(dat, copy=TRUE, blockSize, theta = 1)
```

Arguments

`dat`	TilingFeatureSet
`copy`	Only relevant when using disk-backed objects. If TRUE a copy will be made leaving the original object (dat) unchanged. The input object will not be preserved if copy=FALSE
`blockSize`	The array is divided into a series of blockSize x blockSize rectangular blocks and the average signal level calculated for each. If blockSize is unspecified a size is chosen that gives about 1250 probes per block.
`theta`	smoothing parameter

Details

The array is divided into a set of blockSize x blockSize squares. A kernel smoother is then used to even out spatial artifacts.

Value

a TilingFeatureSet

Author(s)

Martin Aryee <aryee@jhu.edu>

Examples

```
# See normalizeBetweenSamples
```

`validatePd` *Validate a sample description file for two-color microarray data*

Description

Checks a sample description file describing two-color arrays for proper formatting and if requested guesses column numbers for file names, sample labels and group labels.

Usage

```
validatePd(pd, fileNameColumn, sampleNameColumn, groupColumn,
ut = "_532.xys", md = "_635.xys")
```

Arguments

`pd`	A data frame containing the sample description table
`fileNameColumn`	Number or name of column containing file names (optional)
`sampleNameColumn`	Number or name of column containing sample names (optional)
`groupColumn`	Number or name of column containing group labels (optional)
`ut`	the file ending that designates untreated channel files
`md`	the file ending that designates methyl-depleted channel files

Details

This function checks the formatting of a sample description file to make sure it has suitable columns for file names, sample names and (optionally) group labels. The sample description file should have one line per channel, i.e. two lines per sample corresponding to the red and green channel data files. Values in the sample name column are used to pair the two channels together. If fileNameColumn, sampleNameColumn and/or groupColumn are unspecified a guess will be made.

Value

If the input data frame is valid: a list containing the fileNameColumn, sampleNameColumn and groupColumn. If the input data frame is invalid: FALSE

Author(s)

Martin Aryee <aryee@jhu.edu>

See Also

`readCharm`

Examples

```
# See dmrFdr
```

Index