Package 'charm'

October 9, 2013

Type Package

Title Analysis of DNA methylation data from CHARM microarrays

Version 2.6.0

Date 2012-10-17

Author Martin Aryee, Peter Murakami, Harris Jaffee, Rafael Irizarry

Maintainer Peter Murakami <pmurakam@jhsph.edu>

Depends R (>= 2.14.0), Biobase, SQN, fields, RColorBrewer, genefilter

Imports BSgenome, Biobase, oligo (>= 1.11.31), oligoClasses(>= 1.17.39), ff, preprocessCore, methods, stats, Biostrings,IRanges, siggenes, nor1mix, gtools, grDevices, graphics, utils,limma, parallel, sva(>= 3.1.2)

Suggests charmData, BSgenome.Hsapiens.UCSC.hg18, corpcor

Description

This package implements analysis tools for DNA methylation data generated using Nimblegen microarrays and the McrBC protocol. It finds differentially methylated regions between samples, calculates percentage methylation estimates and includes array quality assessment tools.

License LGPL (>= 2)

LazyLoad yes

biocViews Microarray, Bioinformatics, DNAMethylation

R topics documented:

`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
```

`clusterMaker` *Redefine array regions given chromosomal coordinates.*

Description

Redefine array regions given chromosomal coordinates.

Usage

```
clusterMaker(chr,pos,order.it=TRUE,maxGap=300)
```

Arguments

`chr`	Vector of chromosome names
`pos`	Vector of positions within chromosomes.
`order.it`	order probes if they are not already ordered.
`maxGap`	maximum allowable gap between probe start positions for probes to be grouped into the same region.

Details

Redefine array regions given chromosomal coordinates.

Author(s)

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

See Also

`dmrFind`, `plotDMRs`, `plotRegions`

Examples

```
# See qval
```

cmdsplot *Classical multi-dimensional scaling plot of charm data.*

Description

Unsupervised clustering of charm data samples by classical multi-dimensional scaling.

Usage

```
cmdsplot(labcols, expcol, rawData, p, okqc=1:nrow(p), noXorY=TRUE, outfile="./cmds_topN.pdf", topN=c(1
```

Arguments

labcols	vector of colors, one for each group being plotted (each unique value of pData(rawData)[,expcol])
expcol	name of the column of pData(rawData) the values of which the points are colored differently by.
rawData	the TilingFeatureSet object that p came from.
p	the matrix of percentage methylation values (scale: 0, 1) from using methp on rawData. One column per sample
okqc	vector of indices identifying which rows of p to use.
noXorY	logical value for whether or not (TRUE or FALSE) to ignore probes in chr X and Y?
outfile	file argument to pdf()
topN	Use only the

Author(s)

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

See Also

regionPlot, dmrFinder, dmrFdr

Examples

```
# See dmrFdr
```

controlQC *Boxplots of control and non-control probes*

Description

Make boxplots of the non-control probes and the control probes (after spatial and background correction but before any normalization), to confirm that the control probes have a lower distribution of intensities than the non-control probes.

Usage

```
controlQC(rawData, controlProbes=NULL, controlIndex=NULL, IDcol, expcol, ylimits=c(-6,8), outfile="./
```

Arguments

rawData	a TilingFeatureSet object
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) (getContainer reports the container values for the pm probes). This is used only if controlIndex is not provided. Either this or controlIndex must be provided.
controlIndex	a vector of non-CpG control probe indices (can be obtained using getControlIndex). Either this or controlProbes must be provided.
IDcol	column name of pData(rawData) with which to label samples in the output plot.
expcol	name of the column of pData(rawData) by which to order and, if not numeric, color the boxes.
ylimits	ylim argument to plot()
outfile	file argument to pdf()
height	height argument to pdf()
width	width argument to pdf()

Value

A data frame with one row per sample and columns

non_control Median methylation value among the non-control probes

control Median methylation value among the control probes

diff non_control-control

Author(s)

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

See Also

regionPlot, dmrFinder, dmrFdr

Examples

```
# See dmrFdr
```

---

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 (found using the dmrFinder function) using a permutation approach. For differentially methylated regions found using the dmrFind function, use the qval function instead.

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

qval, 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,
                      removeIf=expression(nprobes<4 | abs(diff)<.05 | abs(maxdiff)<.05),
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:
#cpg.cur = read.delim("http://rafalab.jhsph.edu/CGI/model-based-cpg-islands-hg18.txt", as.is=TRUE)
#dmrPlot(dmr=dmr, which.table=1, which.plot=1:5, legend.size=1, all.lines=TRUE, all.points=TRUE, colo

## plot any given genomic regions using this data, supplying the regions in a data frame that must have co
#mytab = data.frame(chr=as.character(c(dmr$tabs[[1]]$chr[1],"chrY",dmr$tabs[[1]]$chr[-1])), start=as.
#regionPlot(tab=mytab, dmr=dmr, cpg.islands=cpg.cur, Genome=Hsapiens, outfile="./myregions.pdf", whic
     ## note that region 2 is not plotted since it is not on the array.

     ## Example of paired analysis:
  pData(rawData)$pair = c(1,2,1,2) ## fake pairing information for this example.
     dmr2 <- dmrFinder(rawData, p=p, groups=grp,
                       compare=c("colon", "liver"),
            removeIf=expression(nprobes<4 | abs(diff)<.05 | abs(maxdiff)<.05),
                       paired=TRUE, pairs=pData(rawData)$pair, cutoff=0.95)

#dmrPlot(dmr=dmr2, which.table=1, which.plot=c(3), legend.size=1, all.lines=TRUE, all.points=TRUE, col
#regionPlot(tab=mytab, dmr=dmr2, cpg.islands=cpg.cur, Genome=Hsapiens, outfile="myregions.pdf", which
}
```

`dmrFind` *Identify DMR candidates using a regression-based approach and correcting for batch effects.*

Description

Identify DMR candidates using a regression-based approach and correcting for batch effects.

Usage

```
dmrFind(p=NULL, logitp=NULL, svs=NULL, mod, mod0, coeff, pns, chr, pos, only.cleanp=FALSE, only.dmrs=FA
```

Arguments

p	matrix of methylation percentage estimates. Either this or logitp must be provided. Can be an ff_matrix object.
logitp	matrix of logit-transformed methylation percentage estimates. Either this or p must be provided. Can be an ff_matrix object.
svs	surrogate variables whose effect will be corrected for. This should be svaobj$sv, where svaobj is the object returned by sva(). Setting svs=0 will result in sva not being used.
mod	The mod argument provided to sva() which yielded svs. This should be a design matrix with all the adjustment covariates and (in the rightmost column(s)) your covariate of interest.
mod0	The mod0 argument provided to sva() which yielded svs. This should be a design matrix with just the adjustment covariates. Thus it should be the same as mod, excluding the rightmost column(s) for the covariate of interest.
coeff	a character or numeric index for the column of mod that identifies the covariate column of interest.
pns	vector of region names for the probes corresponding to rows of p or logitp.
chr	vector of chromosomal identifiers for the probes corresponding to rows of p or logitp.
pos	vector of chromosomal coordinates for the probes corresponding to rows of p or logitp.
only.cleanp	if TRUE, only return the matrix of methylation percentage estimates after removing the batch effects (columns of sv)
only.dmrs	if TRUE, do not return the matrix of methylation percentage estimates that have had the batch effects (columns of sv) removed (called cleanp).
rob	One of the outputs of dmrFind is cleanp, which is the input p matrix after removing batch effects identified by SVA. By default these are the only effects removed from the p matrix. However, if you set rob=FALSE, then the other adjustment variables in mod and mod0 (all other variables besides the covariate of interest) are also removed. This will affect the methylation levels shown in plots using plotDMRs, plotRegions, and any other function that uses the cleanp output of dmrFind. It does not affect the selection of DMRs, though, except in the application of the filter at the end of the function where DMRs with an average difference between the 2 groups (or the average correlation between methylation and the covariate if the covariate is continuous) of less than the min.value argument are filtered out, since this step uses the cleanp matrix to calculate those averages or that correlation.

use.limma	Use the linear modeling approach (borrowing strength across probes) of lmFit in the limma package.
smoo	which method to use for smoothing. "weighted loess", "loess", or "runmed".
k	k argument to runmed() if smoo="runmed".
SPAN	see DELTA. Only used if smoo="loess"
DELTA	span parameter in loess smoothing will = SPAN/(DELTA * number of probes in the plotted region). Only used if smoo="loess".
use	If "sbeta", identify DMRs by segmenting the smoothed effect estimates. If "swald", identify DMRs by segmenting the smoothed wald statistics.
Q	Identify DMRs as the consecutive groups of probes whose smoothed effect estimate (if use="sbeta") or smoothed wald statistics (if use="swald") exceed this quantile.
min.probes	The minimum allowable number of probes in a DMR candidate.
min.value	The minimum allowable average difference in methylation percentage between the 2 groups if covariate is categorical, or the minimum average correlation between methylation and the covariate if covariate is continuous.
keepXY	if FALSE, exclude DMRs in "chrX" and "chrY".
sortBy	column of DMR table to sorty by.
verbose	print progress messages if TRUE.
vfilter	vfilter argument to sva function. The number of most variable probes to use when building SVs–must be between 100 and m, where m is the total number of probes. vfilter=NULL by default, which means vfilter=m. Setting this to something smaller, like 100000, will typically yield satisfactory SVs and is advisable when the size of the data is very large and memory is limited (as when the p and/or logitp arguments given to dmrFind are ff_matrix objects).
nsubsets	used if p or logitp are ff_matrix objects. Rather than doing computations on the whole logitp matrix, break up its m rows into chunks of m/nsubsets rows. Default is 50, but if even that uses too much memory, set this higher. Results do not depend on the value chosen.
...	Additional arguments passed to sva()

Details

Identify DMR candidates using a regression-based approach and correcting for batch effects.

Value

If only.cleanp=TRUE, only the cleanp matrix (below) is returned. Otherwise, the function returns a list with

dmrs	A data frame with all DMR candidates, with columns: **chr** chromosome of DMR **start** start of DMR (bp) **end** end of DMR (bp)

value average value of the smoothed effect estimate within the DMR if use="sbeta" (the default), or the average value of the smoothed wald statistic within the DMR if use="swald"

area nprobes x value

pns name of the region on the array in which the DMR candidate was identified.

indexStart index of first probe in DMR. This indexes chr, pos, pns, and cleanp

indexEnd index of last probe in DMR. This indexes chr, pos, pns, and cleanp

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

avg average (across probes) percentage methylation difference within the DMR if covariate is categorical, or average (across probes) correlation between cleanp and covariate if covariate is continuous.

max maximum (across probes) percentage methylation difference within the DMR if covariate is categorical, or maximum (across probes) correlation between cleanp and covariate if covariate is continuous.

area.raw nprobes x avg

`pval` a vector of p-values for the t-test at each probe (in same order as rows of cleanp)

`pns` a vector of probe region names corresponding to the rows of cleanp

`chr` a vector of chromosomes corresponding to the rows of cleanp

`pos` a vector of positions corresponding to the rows of cleanp

`args` A list containing all the arguments provided to dmrFind. If svs was not provided, svs here will be the surrogate variables obtained from sva.

If only.dmrs=FALSE,

`cleanp` The matrix of percentage methylation estimates, after subtracting batch effects. If rob=FALSE, the effects of the other adjustment covariates are removed also.

If covariate is continuous,

`beta` the effect estimate at each probe.

`sbeta` the smoothed effect estimate at each probe.

Author(s)

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

See Also

`plotDMRs`, `plotRegions`, `qval`

Examples

```
# See qval
```

dmrFinder *Find differentially methylated regions (DMRs)*

Description

Find differentially methylated regions (DMRs) from tiling microarray data. If you want to adjust for covariates (including batch effects estimated using SVA) or if your covariate of interest is continuous, use the dmrFind function.

Usage

```
dmrFinder(eset=NULL, groups, p=NULL, l=NULL, chr=NULL, pos=NULL, pns=NULL,
sdBins=NULL, controlIndex=NULL, controlProbes=NULL, Indexes=NULL,
filter=NULL, package=NULL, ws=7, verbose=TRUE, compare="all",
withinSampleNorm="loess", betweenSampleNorm="quantile",
cutoff=0.995, sortBy="ttarea", removeIf=expression(nprobes<3),
        paired=FALSE, pairs=NULL, DD=NULL, COMPS=NULL, COMPS.names=NULL)
```

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 (which can be obtained using getControlIndex), to be passed on to the methp function if neither the p nor the l arguments are provided, in which case either this or the controlProbes argument must be provided.
controlProbes	names of probe containers corresponding to control probes (e.g., see values returned by getContainer), to be passed on to the methp function if neither the p nor the l arguments are provided, in which case either this or the controlIndex argument must be provided.
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".

`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).

`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`
: Ignore this argument.

`COMPS`
: Ignore this argument.

`COMPS.names`
: Ignore this argument.

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. This indexes the output of dmrFinder, *not* the input.

indexEnd index of last probe in DMR. This indexes the output of dmrFinder, *not* the input.

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

diff average percentage methylation difference within the DMR (i.e., column p1 - column p2), if p or eset arguments are provided. Otherwise, if only l argument is provided, it is the average difference in l (i.e, column m1 - column m2). Prior to version 2.0.1, if paired=TRUE, this was the average l (logit(percentage) methylation if l=NULL) difference within the DMR regardless of whether p or eset were provided.

maxdiff maximum percentage methylation difference within the DMR, if p or eset arguments are provided. Otherwise, if only l argument is provided, it is the maximum difference in l. Prior to version 2.0.1, if paired=TRUE, this was the maximum l (logit(percentage) methylation if l=NULL) difference within the DMR regardless of whether p or eset were provided. Also prior to package version 2.0.1, this column was reported only in absolute value, however, post-version 2.0.1 the sign is retained.

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

`index` a vector identifying which subset of the input probes (i.e. which elements of the input chr, pos, and pns, and rows of the input p and/or l) were used to search for DMRs. The output objects (chr, pos, pns, p, l, etc) are this subset of probes from the input. Therefore, e.g., while tabs$indexStart:tabs$indexEnd indexes the elements or rows of the output objects for each DMR candidate in tabs, index[tabs$indexStart:tabs$indexEnd] indexes the elements or rows of the input objects.

`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

dmrFind, readCharm, methp, dmrFdr

Examples

```
# See dmrFdr
```

---

dmrPlot *Plot differentially methylated regions (DMRs) found using the dmrFinder function.*

Description

Plot differentially methylated regions (DMRs) from tiling microarray data that were identified using the dmrFinder function. To plot DMRs identified using the dmrFind function, use the plotDMRs function.

Usage

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

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.
cpg.islands	a table with columns "chr","start", and "end" for CpG islands to plot in the second panel.
Genome	the BSgenome object for the organism based upon which your array was designed.

plotG	if TRUE, a third panel of each DMR plot will show the difference between the median green channel value (after subtracting probe medians and correcting for gc content) between the 2 groups. If true, dat must be provided.
dat	if plotG=TRUE, this must be provided. It is the TilingFeatureSet object used when estimating the matrix of percent methylation estimates used in dmrFinder when it produced dmr.
buffer	Number of base pairs to plot on either side of each DMR candidate.
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). The second panel shows the location of CpG's with ticks on the bottom (islands are colored) and the location of mcrbc recognition sites with ticks on the top.

Author(s)

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

See Also

plotDMRs, 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 = NULL, 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
```

---

maxDensity *Find the mode of a density function*

Description

Calculate a density function and find the max point

Usage

```
maxDensity(x, n.pts = 2^14, minPoints=30)
```

Arguments

x	a data vector
n.pts	Number of points to use in density estimation
minPoints	Minimum number of data points to accept

Details

This function finds the maximum of a density function. It is identical to the (unexported) max.density function in affy except that it returns an NA if the number of data points provided is less than `minPoints`

Value

a numeric value

Author(s)

Martin Aryee <aryee@jhu.edu>

Examples

```
x <- rnorm(1000)
maxDensity(x)
```

`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 = NULL, 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". "loess" uses the control-probe loess procedure described in Aryee et al., 2011 (PMID: 20858772).

`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(s) assigned to non-CpG control probes in the array design file, i.e. in the container column of the .ndf design file for your array. The getContainer function from the oligo package can be used do see the container values for all pm probes. This argument is used only if controlIndex is not provided. Either this or controlIndex must be provided.

`controlIndex` a numeric vector of probe indices indicating which pm probes are the non-CpG control probes (can be obtained using getControlIndex). Either this or controlProbes must be provided.

`excludeIndex` a numeric vector of probe indices indicating which pm probes to ignore when creating the between-array normalization target distributions. These probes are also removed from all the plots created if plotDensity is specified, however be aware that they ARE still returned in the output of methp (just without having had any between-array normalization applied to them).

`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. If the dat argument contains ff rather than matrix objects (as when it is produced by readCharm after loading the ff package), the output will be an ff object.

Author(s)

Martin Aryee <aryee@jhu.edu>

See Also

readCharm

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)
}
```

`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=NULL, 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. This function implements the control probe loess procedure described in Aryee et al., 2011 (PMID: 20858772).

Usage

```
normalizeWithinSamples(dat, copy=TRUE,
method = "loess", scale=c(0.99, 0.99),
controlProbes = NULL, 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, excludeIndex = 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.
excludeIndex	a numeric vector of probe indices indicating which pm probes to ignore when plotting.

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)
               ## Not run:
#plotDensity(rawData, controlIndex=ctrlIdx, pdfFile="density.pdf")
}
```

plotDMRs *Plot differentially methylated regions (DMRs) found using the dmrFind function.*

Description

Plot differentially methylated regions (DMRs) from tiling microarray data that were identified using the dmrFind function.

Usage

```
plotDMRs(dmrs, Genome, cpg.islands, exposure, outfile, which_plot=1:50, which_lines=NULL, which_points
```

Arguments

dmrs	a list object as returned by dmrFind.
Genome	the BSgenome object for the organism based upon which your array was designed.
cpg.islands	a table with columns "chr","start", and "end" for CpG islands to plot in the second panel.
exposure	The covariate of interest.
outfile	the name of the file to save (including the full path)
which_plot	numeric vector of indices identifying which DMR candidates from dmrs$dmrs to plot.
which_lines	vector specifying which groups (unique elements of exposure) to plot the lines for. If NULL (the default), plots lines for all groups. Only applies if exposure is categorical.
which_points	vector specifying which groups (unique elements of exposure) to plot the points for. If NULL (the default), plots points for all groups. Only applies if exposure is categorical.
SPAN	see DELTA. Only used if smoo="loess"
DELTA	span parameter in loess smoothing will = SPAN/(DELTA * number of probes in the plotted region). Only used if smoo="loess".
smoo	"loess" for loess smoother or "runmed" for running median smoother (runmed with k=3). This does not need to be the same as the smoo argument to dmrFind.
ADD	Number of base pairs to plot on either side of each DMR candidate (if it is covered on the array).
cols	vector of colors to use, one for each group (if covariate is categorical)
point.info	if TRUE, function returns a table identifying which sample is plotted with which number or letter (if pch.groups=NULL, the default).
legend.size	magnification factor for the legend
pch.groups	vector whose length is equal to the number of samples. Each unique value will be plotted with a different point type.

panel3	if panel3="G", the third panel of each DMR plot will show the difference between the median green channel value (after subtracting probe medians and correcting for gc content) between the 2 groups (i.e., the group defined by mod[,coef] in dmrFind minus the reference group). If panel="G", seq argument must be provided. If panel!="G", the 3rd panel will show -log10(dmrs$pval). G
G	matrix of green channel intensities to use for plotting in the 3rd panel if panel3="G".
seq	vector of probe sequences corresponding to the rows of G (and dmrs$cleanp) if panel3="G".

Details

This function plots the differentially methylated regions (DMRs) that were identified using the dmrFind function.

Author(s)

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

See Also

plotRegions, dmrFind, qval

Examples

```
# See qval
```

---

plotRegions *Plot user-provided regions.*

Description

Plot user-provided regions.

Usage

```
plotRegions(thetable, cleanp, chr, pos, seq=NULL, Genome, cpg.islands, exposure, exposure.continuous=F
```

Arguments

thetable	a table with columns for "chr","start", and "end", identifying the genomic regions to plot (if they are covered on the array).
cleanp	the matrix of percent methylation estimates to be used for plotting
chr	vector of chromosome labels for the probes in cleanp
pos	vector of chromosomal coordinates for the probes in cleanp
seq	vector of probe sequences corresponding to the rows of G (and cleanp). Needed only if panel3="G".

Genome	the BSgenome object for the organism based upon which your array was designed.
cpg.islands	a table with columns "chr","start", and "end" for CpG islands to plot in the second panel.
outfile	the name of the file to save (including the full path)
exposure	The covariate of interest.
exposure.continuous	set to TRUE if exposure is a continuous variable.
which_lines	vector specifying which groups (unique elements of exposure) to plot the lines for. If NULL (the default), plots lines for all groups. Only applies if exposure is categorical.
which_points	vector specifying which groups (unique elements of exposure) to plot the points for. If NULL (the default), plots points for all groups. Only applies if exposure is categorical.
ADD	Number of base pairs to plot on either side of each DMR candidate (if it is covered on the array).
cols	vector of colors to use, one for each group (if covariate is categorical)
legend.size	magnification factor for the legend
smoo	"loess" for loess smoother or "runmed" for running median smoother (runmed with k=3).
SPAN	see DELTA. Only used if smoo="loess"
DELTA	span parameter in loess smoothing will = SPAN/(DELTA * number of probes in the plotted region). Only used if smoo="loess".
panel3	if panel3="G", the third panel of each DMR plot will show the difference between the median green channel value (after subtracting probe medians and correcting for gc content) between the 2 groups (i.e., the group defined by mod[,coef] in dmrFind minus the reference group). If panel="G", seq argument must be provided. If panel!="G", the 3rd panel will show -log10(dmrs$pval).
G	matrix of green channel intensities to use for plotting in the 3rd panel if panel3="G".
grs	if panel3="G", plot difference between the median green channel value (after subtracting probe medians and correcting for gc content) between these 2 groups.

Details

This function plots user-provided regions.

Author(s)

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

See Also

plotDMRs, dmrFind, qval

Examples

```
# See qval
```

`pmQuality` *Calculate probe quality scores*

Description

pmQuality calculates probe quality for each pm probe by comparing the total DNA signal (green) to the distribution of the background probe signals. 0 means lower than all background probes. 100 means higher than all background probes.

Usage

```
pmQuality(dat, channel="channel1", verbose=FALSE, idx=NULL)
```

Arguments

`dat`	a TilingFeatureSet object
`channel`	which channel to assess
`verbose`	logical value for whether or not to print message to screen
`idx`	vector of probe indices specifying which probes to evaluate. By default all pm probes are used.

Author(s)

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

See Also

`regionPlot`, `dmrFinder`, `dmrFdr`

Examples

```
# See dmrFdr
```

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)
               ## Not run:
#qcReport(rawData, file="qcReport.pdf")
}
```

---

qval *Obtain False Discovery Rate q-values for the DMR candidates returned by dmrFind.*

Description

Obtain False Discovery Rate q-values for the DMR candidates returned by dmrFind.

Usage

```
qval(p=NULL, logitp=NULL, dmr, numiter=500, seed=54256, verbose=FALSE, mc=1, return.permutations=FALSE
```

Arguments

p	The matrix of methylation percentage estimates provided to dmrFind (argument p) when it returned dmr, if p was provided.
logitp	The matrix of logit-transformed methylation percentage estimates provided to dmrFind (argument logitp) when it returned dmr, if logitp was provided.
dmr	The output of dmrFind with the dmr table to obtain q-values for.
numiter	The number of times to iterate the resampling procedure.
seed	random seed.
verbose	verbose argument to dmrFind.
mc	Number of CPUs to use for parallel processing.
return.permutations	if TRUE, also return a matrix defining the resampling at each iteration.
method	if "direct" (the default), the resulting table will have an additional column ("qvalue") with the direct estimate of the FDR qvalue as described below. If "pool", the resulting table will have additional columns ("pvalue.pool" and "qvalue.pool") with the p-values and corresponding q-values obtained as the proportion of statistics in the null distribution that are greater than or equal to the observed

statistic, where the null distribution is obtained by simply pooling together all the statistic vectors from all iterations. If "fwer", the resulting table will have an additional column ("pvalue.fwer") with the p-value adjusted for control of the family-wise error rate (obtained as the proportion of maximum statistics (across all iterations) that are greater than or equal to the observed statistic). More than one value for method may be specified.

`fwer.num` positive integer vector, needed if method argument includes "fwer". For each number, a separate output column will be returned with the FWER(n)-controlling adjusted p-values, where FWER(n) is defined as the probability of making n or more false discoveries (type I errors).

Details

The q-value for region i is estimated by a resampling procedure. It is estimated as x/N, where N is the number of dmrs in the DMR table returned by dmrFind whose statistic (defined by the sortBy argument to dmrFind) is >= the statistic for region i, and x is the average (across all iterations' DMR tables) number of DMRs with statistic >= the statistic for region i. Monotonicity is then enforced and q-values are capped at 1. For example, to estimate the q-value for the 5th dmr in your dmr table that you ranked by area, N is 5. If it has area=100, then x is the average number of dmrs with area>=100, averaging over all the tables produced by random-resampling iterations. To the extent that the proportion of the genome covered by the array that is truly non-differentially methylated is less than 1, x will slightly overestimate the true number of false DMR candidates expected to be returned from the data, and hence the q-value will be slightly conservative.

The resampling procedure used to obtain null data for each iteration is as follows. 1. Add the surrogate variables identified by SVA to the null model defined by the mod0 argument to dmrFind, and obtain fitted methylation values M, where rows correspond to probes and columns correspond to arrays. 2. Add the surrogate variables identified by SVA to the full model defined by the mod argument to dmrFind, and obtain residuals R, where rows correspond to probes and columns correspond to arrays. Then, for iteration k, obtain a column-resampled version of R, Q (resampling with replacement), and obtain the methylation values to be used in iteration k as M+Q.

Value

the dmr table in your dmr object (dmr$dmrs), with an additional column ("qvalue") for the Fdr q-value (if method includes "direct"), additional columns for the pooled estimates of the p-values and q-values (if method includes "pool"), and an additional column for the FWER-adjusted p-values (if method includes "fwer") (see method argument). There is also a column called qvalue0 which is just the qvalue column before enforcing monotonicity.

Author(s)

Martin Aryee, Peter Murakami <pmurakam@jhsph.edu>, Rafael Irizarry

See Also

`dmrFind`, `plotDMRs`, `plotRegions`

Examples

```
## See vignette.
if (require(charmData) & require(BSgenome.Hsapiens.UCSC.hg18)) {
    dataDir <- system.file("data", package="charmData")
    phenodataDir <- system.file("extdata", package="charmData")
    pd <- read.delim(file.path(phenodataDir, "phenodata.txt"))
    res <- validatePd(pd)

    ## Read in raw data:
 rawData <- readCharm(files=pd$filename, path=dataDir, sampleKey=pd, sampleNames=pd$sampleID)
    ## Check quality of arrays:
    #qual <- qcReport(rawData, file="qcReport.pdf")

    ## Assess individual probe qualities:
    pmq = pmQuality(rawData)
    rmpmq = rowMeans(pmq)
    okqc = which(rmpmq>75)

 ## Identify control probes as the probes at positions surrounded by a CpG-free 600bp window:
    ctrlIdx <- getControlIndex(rawData, subject=Hsapiens, noCpGWindow=600)

 ## Check that these control probes do indeed have lower intensities than the non-control probes (after spatial
 #controlQC(rawData=rawData, controlIndex=ctrlIdx, IDcol="sampleID", expcol="tissue", ylimits=c(-6,8), outfi

    chr = pmChr(rawData)
    pns = probeNames(rawData)
    pos = pmPosition(rawData)
    seq = pmSequence(rawData)
    pd  = pData(rawData)

    ## Estimate percent methylation:
    p <- methp(rawData, controlIndex=ctrlIdx, plotDensityGroups=pd$tissue)

    ## unsupervised clustering of samples:
 #cmdsplot(labcols=c("red","black","blue"), expcol="tissue", rawData=rawData, p=p, okqc=okqc, noXorY=TRUE, c

 ## Do not look for DMRs among control probes or probes with average probe quality score less than or equal to 75
    Index=setdiff(which(rmpmq>75),ctrlIdx)
    Index = Index[order(chr[Index], pos[Index])]
    p = p[Index,]
    seq = seq[Index]
    chr = chr[Index]
    pos = pos[Index]
    pns = pns[Index]
    pns=clusterMaker(chr,pos)

 ## Identify DMR candidates between colon and liver (in this example not adjusting for any other covariates (bes
    mod0 = matrix(1,nrow=nrow(pd),ncol=1)
    mod  = model.matrix(~1 +factor(pd$tissue,levels=c("liver","colon","spleen")))
    thedmrs = dmrFind(p=p, mod=mod, mod0=mod0, coeff=2, pns=pns, chr=chr, pos=pos)

 ## Obtain FDR q-values for each DMR candidate. In practice, numiter should be set much higher.
```

```
   withq = qval(p=p, dmr=thedmrs, numiter=2, verbose=FALSE, mc=1)

   #### Plotting not run:
## Plot DMR candidates 1,2, and 4, for example.  First have to load a table of CpG islands.
cpg.cur = read.delim("http://rafalab.jhsph.edu/CGI/model-based-cpg-islands-hg18.txt", as.is=TRUE)
#plotDMRs(dmrs=thedmrs, Genome=Hsapiens, cpg.islands=cpg.cur, exposure=pd$tissue, outfile="./colon-liver.p

## Plot DMR candidates, and in the 3rd panel plot the difference in average green channel:
   dat0 = spatialAdjust(rawData, copy=FALSE)
   dat0 = bgAdjust(dat0, copy=FALSE)
   G = pm(dat0)[,,1] #from oligo
   G = G[Index,]
#plotDMRs(dmrs=thedmrs, Genome=Hsapiens, cpg.islands=cpg.cur, exposure=pd$tissue, outfile="./colon-liver2.

   ## Example if covariate of interest is continuous:
   pd$x = c(1,2,3,4,5,6)
   mod0 = matrix(1,nrow=nrow(pd),ncol=1)
   mod  = model.matrix(~1 +pd$x)
   coeff = 2
 thedmrs2 = dmrFind(p=p, mod=mod, mod0=mod0, coeff=coeff, pns=pns, chr=chr, pos=pos)

## If covariate of interest is continuous, you can still plot it like it is categorical by categorizing it for p
groups = as.numeric(cut(mod[,coeff],c(-Inf,2,4,Inf))) #You can change these cutpoints.
   pd$groups = c("low","medium","high")[groups]
#plotDMRs(dmrs=thedmrs2, Genome=Hsapiens, cpg.islands=cpg.cur, exposure=pd$groups, outfile="./test.pdf", wh

## Otherwise, if covariate of interest is continuous, plot will show correlation with covariate:
#plotDMRs(dmrs=thedmrs2, Genome=Hsapiens, cpg.islands=cpg.cur, exposure=pd$x, outfile="./x.pdf", which_plot

   ## Plot arbitrary regions:
   mytable = thedmrs$dmrs[,c("chr","start","end")]
   mytable[2,] = c("chr1",1,1000) #not on array
   mytable$start = as.numeric(mytable$start)
   mytable$end = as.numeric(mytable$end)
 #plotRegions(thetable=mytable[1:5,], cleanp=thedmrs$cleanp, chr=chr, pos=pos, Genome=Hsapiens, cpg.islands=
}
```

---

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. If the ff package is loaded before using this function, the output will contain ff rather than matrix objects.

Author(s)

Martin Aryee <aryee@jhu.edu>

References

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

See Also

methp, dmrFinder

Examples

```
# See normalizeBetweenSamples
```

`regionMatch` *Given two data frames with columns "chr", "start", and "end", identify the nearest region in one to the other.*

Description

Given two data frames with columns "chr", "start", and "end", identify the nearest region in one (object2) to the other (object1).

Usage

```
regionMatch(object1, object2, verbose=TRUE)
```

Arguments

`object1`	Data frame with columns "chr", "start", and "end". For each of the rows of this data frame, find the nearest regions in object2.
`object2`	Data frame with columns "chr", "start", and "end".
`verbose`	print progress.

Details

Given two data frames with columns "chr", "start", and "end", identify the nearest region in one (object2) to the other (object1).

Value

A data frame with one row corresponding for each row of object1, and columns:

dist bp separating the region in object1 and the nearest region in object2

matchIndex row index of object2 for the region that is closest to the region in object1

type "inside" if nearest region in object2 is wholly contained inside the region in object 1, "cover" if nearest region in object2 covers the whole region in object1, "disjoint" if there is no overlap between the region in object1 and the nearest region in object1, and "overlap" if the region in object1 and the nearest region in object2 overlap but one does not wholly cover the other.

amountOverlap amount of overlap between the region in object1 and the nearest region in object2

insideDist for type="inside" regions, the smaller (fewest bp) of 1., the end position of region in object1 - the end position of the nearest region in object2, and 2., the start position of region in object1 - the start position of the nearest region in object2.

size1 number of bp in region in object1

size2 number of bp in nearest region in object2

Author(s)

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

See Also

dmrFind, plotDMRs, plotRegions

Examples

```
# See qval
```

---

regionPlot *Plot user-supplied genomic regions using data returned by the dmrFinder function.*

Description

Plot any given genomic regions from tiling microarray data using data returned by the dmrFinder function.

Usage

```
regionPlot(tab, dmr, cpg.islands, Genome, outfile="./regions.pdf", which.plot=1:10, plot.these, cl, le
```

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.
cpg.islands	a table with columns "chr","start", and "end" for CpG islands to plot in the second panel.
Genome	the BSgenome object for the organism based upon which your array was designed.
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.
plot.these	if dmr results are from unpaired analysis, specify the groups to plot. Default is colnames(dmr$gm). If dmr results are from paired analysis, specify the comparisons to plot. Default is colnames(dmr$sMD).
cl	a vector of line and point colors, one for each element of plot.these 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.

plotG	if TRUE, a third panel of each DMR plot will show the difference between the median green channel value (after subtracting probe medians and correcting for gc content) between the 2 groups. If true, dat must be provided.
dat	if plotG=TRUE, this must be provided. It is the TilingFeatureSet object used when estimating the matrix of percent methylation estimates used in dmrFinder when it produced dmr.
grs	if plotG=TRUE, this must be provided. It is the names of the two groups whose difference in G should be plotted in the 3rd panel.

Details

This function enables plotting of any regions, not just DMRs, using the results of dmrFinder. The second panel shows the location of CpG's with ticks on the bottom (islands are colored) and the location of mcrbc recognition sites with ticks on the top.

Author(s)

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

See Also

plotRegions, 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