Package ‘minfi’

May 15, 2025

Version 1.54.1

Title Analyze Illumina Infinium DNA methylation arrays

Description Tools to analyze & visualize Illumina Infinium methylation arrays.

Depends methods, BiocGenerics (>= 0.15.3), GenomicRanges, SummarizedExperiment (>= 1.1.6), Biostrings, bumphunter (>= 1.1.9)

Suggests IlluminaHumanMethylation450kmanifest (>= 0.2.0), IlluminaHumanMethylation450kanno.ilmn12.hg19 (>= 0.2.1), minfiData (>= 0.18.0), minfiDataEPIC, FlowSorted.Blood.450k (>= 1.0.1), RUnit, digest, BiocStyle, knitr, rmarkdown, tools

Imports S4Vectors, GenomeInfoDb, Biobase (>= 2.33.2), IRanges, beanplot, RColorBrewer, lattice, nor1mix, siggenes, limma, preprocessCore, illuminaio (>= 0.23.2), DelayedMatrixStats (>= 1.3.4), mclust, genefilter, nlme, reshape, MASS, quadprog, data.table, GEOquery, stats, grDevices, graphics, utils, DelayedArray (>= 0.15.16), HDF5Array, BiocParallel

Collate AllGenerics.R MethylSet-class.R RatioSet-class.R RGChannelSet-class.R RGChannelSetExtended-class.R GenomicMethylSet-class.R GenomicRatioSet-class.R eSet_methods.R utils.R IlluminaMethylationManifest-class.R IlluminaMethylationAnnotation-class.R minfiQC.R getSex.R dmpFinder.R blocks.R plot.R plotBetasByType.R preprocessRaw.R preprocessIllumina.R detectionP.R preprocessSwan.R preprocessQuantile.R preprocessNoob.R preprocessFunnorm.R qc.R read.450k.R read.meth.R read.meth2.R read.geo.R read.manifest.R bumphunter.R estimateCellCounts.R gaphunter.R compartments.R combineArrays.R DelayedArray_utils.R

VignetteBuilder knitr

License Artistic-2.0

URL https://github.com/hansenlab/minfi

BugReports https://github.com/hansenlab/minfi/issues

LazyData yes

biocViews ImmunoOncology, DNAMethylation, DifferentialMethylation, Epigenetics, Microarray, MethylationArray, MultiChannel, TwoChannel, DataImport, Normalization, Preprocessing, QualityControl

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

git_branch RELEASE_3_21

git_last_commit ff98799

git_last_commit_date 2025-05-08

Repository Bioconductor 3.21

Date/Publication 2025-05-14

Author Kasper Daniel Hansen [cre, aut],
Martin Aryee [aut],
Rafael A. Irizarry [aut],
Andrew E. Jaffe [ctb],
Jovana Maksimovic [ctb],
E. Andres Houseman [ctb],
Jean-Philippe Fortin [ctb],
Tim Triche [ctb],
Shan V. Andrews [ctb],
Peter F. Hickey [ctb]

Maintainer Kasper Daniel Hansen <kasperdanielhansen@gmail.com>

Contents

`minfi-package` *Analyze Illumina's methylation arrays*

Description

Tools for analyzing and visualizing Illumina methylation array data. There is special focus on the 450k array; the 27k array is not supported at the moment.

Details

The package contains a (hopefully) useful vignette; this vignette contains a lengthy description of the package content and capabilities.

blockFinder *Finds blocks of methylation differences for Illumina methylation arrays*

Description

Finds blocks (large scale regions) of methylation differences for Illumina methylation arrays

Usage

```
blockFinder(object, design, coef = 2, what = c("Beta", "M"),
                        cluster = NULL, cutoff = NULL,
                        pickCutoff = FALSE, pickCutoffQ = 0.99,
                        nullMethod = c("permutation","bootstrap"),
                        smooth = TRUE, smoothFunction = locfitByCluster,
                        B = ncol(permutations), permutations = NULL,
                        verbose = TRUE, bpSpan = 2.5*10^5,...)
```

Arguments

`object`	An object of class GenomicRatioSet.
`design`	Design matrix with rows representing samples and columns representing covariates. Regression is applied to each row of mat.
`coef`	An integer denoting the column of the design matrix containing the covariate of interest. The hunt for bumps will be only be done for the estimate of this coefficient.
`what`	Should blockfinding be performed on M-values or Beta values?
`cluster`	The clusters of locations that are to be analyzed together. In the case of microarrays, the clusters are many times supplied by the manufacturer. If not available the function `clusterMaker` can be used to cluster nearby locations.
`cutoff`	A numeric value. Values of the estimate of the genomic profile above the cutoff or below the negative of the cutoff will be used as candidate regions. It is possible to give two separate values (upper and lower bounds). If one value is given, the lower bound is minus the value.
`pickCutoff`	Should a cutoff be picked automatically?
`pickCutoffQ`	The quantile used for picking the cutoff using the permutation distribution.
`nullMethod`	Method used to generate null candidate regions, must be one of 'bootstrap' or 'permutation' (defaults to 'permutation'). However, if covariates in addition to the outcome of interest are included in the design matrix (ncol(design)>2), the 'permutation' approach is not recommended. See vignette and original paper for more information.
`smooth`	A logical value. If TRUE the estimated profile will be smoothed with the smoother defined by `smoothFunction`

`smoothFunction`	A function to be used for smoothing the estimate of the genomic profile. Two functions are provided by the package: `loessByCluster` and `runmedByCluster`.
`B`	An integer denoting the number of resamples to use when computing null distributions. This defaults to 0. If `permutations` is supplied that defines the number of permutations/bootstraps and `B` is ignored.
`permutations`	is a matrix with columns providing indexes to be used to scramble the data and create a null distribution. If this matrix is not supplied and `B`>0 then these indexes created using the function `sample`.
`verbose`	Should the function be verbose?
`bpSpan`	Smoothing span. Note that this defaults to a large value becuase we are searching for large scale changes.
`...`	further arguments sent to `bumphunterEngine`.

Details

The approximately 170,000 open sea probes on the 450k can be used to detect long-range changes in methylation status. These large scale changes that can range up to several Mb have typically been identified only through whole-genome bisulfite sequencing. `blockFinder` groups the averaged methylation values in open-sea probe clusters (See cpgCollapse) into large regions in which the bumphunter procedure is applied with a large (250KB+) smoothing window.

Note that estimating the precise boundaries of these blocks are constrained by the resolution of the array.

Value

FIXME

See Also

`cpgCollapse`, and `bumphunter`

`bumphunter-methods` *Methods for function* `bumphunter` *in Package* **minfi**

Description

Estimate regions for which a genomic profile deviates from its baseline value. Originally implemented to detect differentially methylated genomic regions between two populations, but can be applied to any CpG-level coefficient of interest.

Usage

```
## S4 method for signature 'GenomicRatioSet'
bumphunter(object, design, cluster=NULL,
          coef=2,  cutoff=NULL, pickCutoff=FALSE, pickCutoffQ=0.99,
          maxGap=500,  nullMethod=c("permutation","bootstrap"),
          smooth=FALSE, smoothFunction=locfitByCluster,
          useWeights=FALSE,   B=ncol(permutations), permutations=NULL,
          verbose=TRUE, type = c("Beta","M"), ...)
```

Arguments

`object`	An object of class GenomicRatioSet.
`design`	Design matrix with rows representing samples and columns representing covariates. Regression is applied to each row of mat.
`cluster`	The clusters of locations that are to be analyzed together. In the case of microarrays, the clusters are many times supplied by the manufacturer. If not available the function `clusterMaker` can be used to cluster nearby locations.
`coef`	An integer denoting the column of the design matrix containing the covariate of interest. The hunt for bumps will be only be done for the estimate of this coefficient.
`cutoff`	A numeric value. Values of the estimate of the genomic profile above the cutoff or below the negative of the cutoff will be used as candidate regions. It is possible to give two separate values (upper and lower bounds). If one value is given, the lower bound is minus the value.
`pickCutoff`	Should bumphunter attempt to pick a cutoff using the permutation distribution?
`pickCutoffQ`	The quantile used for picking the cutoff using the permutation distribution.
`maxGap`	If cluster is not provided this maximum location gap will be used to define cluster via the `clusterMaker` function.
`nullMethod`	Method used to generate null candidate regions, must be one of 'boots trap' or 'permutation' (defaults to 'permutation'). However, if covariates in addition to the outcome of interest are included in the design matrix (ncol(design)>2), the 'permutation' approach is not recommended. See vignette and original paper for more information.
`smooth`	A logical value. If TRUE the estimated profile will be smoothed with the smoother defined by `smoothFunction`
`smoothFunction`	A function to be used for smoothing the estimate of the genomic profile. Two functions are provided by the package: `loessByCluster` and `runmedByCluster`.
`useWeights`	A logical value. If `TRUE` then the standard errors of the point-wise estimates of the profile function will be used as weights in the loess smoother `loessByCluster`. If the `runmedByCluster` smoother is used this argument is ignored.
`B`	An integer denoting the number of resamples to use when computing null distributions. This defaults to 0. If `permutations` is supplied that defines the number of permutations/bootstraps and `B` is ignored.

`permutations`	is a matrix with columns providing indexes to be used to scramble the data and create a null distribution when `nullMethod` is set to permutations. If the bootstrap approach is used this argument is ignored. If this matrix is not supplied and `B`>0 then these indexes are created using the function `sample`.
`verbose`	logical value. If `TRUE`, it writes out some messages indicating progress. If `FALSE` nothing should be printed.
`type`	Should bumphunting be performed on M-values ("M") or Beta values ("Beta")?
`...`	further arguments to be passed to the smoother functions.

Details

See help file for `bumphunter` method in the `bumphunter` package for for details.

Value

An object of class `bumps` with the following components:

`tab`	The table with candidate regions and annotation for these.
`coef`	The single loci coefficients.
`fitted`	The estimated genomic profile used to determine the regions.
`pvaluesMarginal`	marginal p-value for each genomic location.
`null`	The null distribution.
`algorithm`	details on the algorithm.

Author(s)

Rafael A. Irizarry, Martin J. Aryee and Kasper D. Hansen

References

AE Jaffe, P Murakami, H Lee, JT Leek, MD Fallin, AP Feinberg, and RA Irizarry. *Bump hunting to identify differentially methylated regions in epigenetic epidemiology studies.* International Journal of Epidemiology (2012) 41(1):200-209. doi:10.1093/ije/dyr238

See Also

`bumphunter`

Examples

```
if(require(minfiData)) {
  gmSet <- preprocessQuantile(MsetEx)
  design <- model.matrix(~ gmSet$status)
  bumps <- bumphunter(gmSet, design = design, B = 0,
                      type = "Beta", cutoff = 0.25)
}
```

combineArrays *A method for combining different types of methylation arrays into a virtual array.*

Description

A method for combining different types of methylation arrays into a virtual array. The three generations of Illumina methylation arrays are supported: the 27k, the 450k and the EPIC arrays. Specifically, the 450k array and the EPIC array share many probes in common. This function combines data from the two different array types and outputs a data object of the user-specified type. Essentially, this new object will be like (for example) an EPIC array with many probes missing.

Usage

```
## S4 method for signature 'RGChannelSet,RGChannelSet'
combineArrays(object1, object2,
                  outType = c("IlluminaHumanMethylation450k",
                              "IlluminaHumanMethylationEPIC"),
                  verbose = TRUE)
## S4 method for signature 'MethylSet,MethylSet'
combineArrays(object1, object2,
                  outType = c("IlluminaHumanMethylation450k",
                              "IlluminaHumanMethylationEPIC",
                              "IlluminaHumanMethylation27k"),
                  verbose = TRUE)
## S4 method for signature 'RatioSet,RatioSet'
combineArrays(object1, object2,
                  outType = c("IlluminaHumanMethylation450k",
                              "IlluminaHumanMethylationEPIC",
                              "IlluminaHumanMethylation27k"),
                  verbose = TRUE)
## S4 method for signature 'GenomicMethylSet,GenomicMethylSet'
combineArrays(object1, object2,
                  outType = c("IlluminaHumanMethylation450k",
                              "IlluminaHumanMethylationEPIC",
                              "IlluminaHumanMethylation27k"),
                  verbose = TRUE)
## S4 method for signature 'GenomicRatioSet,GenomicRatioSet'
combineArrays(object1, object2,
                  outType = c("IlluminaHumanMethylation450k",
                              "IlluminaHumanMethylationEPIC",
                              "IlluminaHumanMethylation27k"),
                  verbose = TRUE)
```

Arguments

object1	The first object.

object2	The second object.
outType	The array type of the output.
verbose	Should the function be verbose?

Details

FIXME: describe the RCChannelSet combination.

Value

The output object has the same class as the two input objects, that is either an RGChannelSet, a MethylSet, a RatioSet, a GemomicMethylSet or a GenomicRatioSet, with the type of the array given by the outType argument.

Author(s)

Jean-Philippe Fortin and Kasper D. Hansen.

Examples

```
if(require(minfiData) && require(minfiDataEPIC)) {
  data(RGsetEx.sub)
  data(RGsetEPIC)
  rgSet <- combineArrays(RGsetEPIC, RGsetEx.sub)
  rgSet
}
```

---

compartments — *Estimates A/B compartments from Illumina methylation arrays*

Description

Estimates A/B compartments as revealed by Hi-C by computing the first eigenvector on a binned probe correlation matrix.

Usage

```
compartments(object, resolution=100*1000, what = "OpenSea", chr="chr22",
                    method = c("pearson", "spearman"), keep=TRUE)
```

Arguments

object	An object of class (Genomic)MethylSet or (Genomic)RatioSet
resolution	An integer specifying the binning resolution
what	Which subset of probes should be used?
chr	The chromosome to be analyzed.
method	Method of correlation.
keep	Should the correlation matrix be stored or not?

Details

This function extracts A/B compartments from Illumina methylation microarrays. Analysis of Hi-C data has shown that the genome can be divided into two compartments (A/B compartments) that are cell-type specific and are associated with open and closed chromatin respectively. The approximately 170,000 open sea probes on the 450k array can be used to estimate these compartments by computing the first eigenvector on a binned correlation matrix. The binning resolution can be specified by `resolution`, and by default is set to a 100 kb. We do not recommend higher resolutions because of the low-resolution probe design of the 450k array.

Value

an object of class `GRanges` containing the correlation matrix, the compartment eigenvector and the compartment labels (A or B) as metadata.

Author(s)

Jean-Philippe Fortin `<jfortin@jhsph.edu>`, Kasper D. Hansen `<kasperdanielhansen@gmail.com>`

References

JP Fortin and KD Hansen. *Reconstructing A/B compartments as revealed by Hi-C using long-range correlations in epigenetic data*. bioRxiv (2015). doi:10.1101/019000.

Examples

```
if (require(minfiData)) {
  GMset <- mapToGenome(MsetEx)
  ## compartments at 1MB resolution; we recommend 100kb.
  comps <- compartments(GMset, res = 10^6)
}
```

`controlStripPlot` *Plot control probe signals.*

Description

Strip plots are produced for each control probe type specified.

Usage

```
controlStripPlot(rgSet, controls = c("BISULFITE CONVERSION I",
    "BISULFITE CONVERSION II"), sampNames = NULL, xlim = c(5, 17))
```

Arguments

`rgSet`	An `RGChannelSet`.
`controls`	A vector of control probe types to plot.
`sampNames`	Sample names to be used for labels.
`xlim`	x-axis limits.

Details

This function produces the control probe signal plot component of the QC report.

Value

No return value. Plots are produced as a side-effect.

Author(s)

Martin Aryee <aryee@jhu.edu>.

See Also

qcReport, mdsPlot, densityPlot, densityBeanPlot

Examples

```
if (require(minfiData)) {

names <- pData(RGsetEx)$Sample_Name
controlStripPlot(RGsetEx, controls=c("BISULFITE CONVERSION I"), sampNames=names)

}
```

---

convertArray — *A method for converting a type of methylation arrays into a virtual array of another type.*

Description

A method for converting a type of methylation array into a array of another type. The three generations of Illumina methylation arrays are supported: the 27k, the 450k and the EPIC arrays. Specifically, the 450k array and the EPIC array share many probes in common. For `RGChannelSet`, this function will convert an EPIC array into a 450k array (or vice-versa) by dropping probes that differ between the two arrays. Because most of the probes on the 27k array have a different chemistry than the 450k and EPIC probes, converting an 27k `RGChannelSet` into another array is not supported. Each array can be converted into another array at the CpG site level, that is any `MethylSet` and `RatioSet` (or `GenomicMethylSet` and `GenomicRatioSet`) can be converted to a 27k, 450k or EPIC array. The output array is specified by the `outType` argument.

Usage

```
## S4 method for signature 'RGChannelSet'
convertArray(object,
                  outType = c("IlluminaHumanMethylation450k",
                              "IlluminaHumanMethylationEPIC"),
                  verbose = TRUE)
## S4 method for signature 'MethylSet'
```

```
convertArray(object,
                  outType = c("IlluminaHumanMethylation450k",
                              "IlluminaHumanMethylationEPIC",
                              "IlluminaHumanMethylation27k"),
                  verbose = TRUE)
## S4 method for signature 'RatioSet'
convertArray(object,
                  outType = c("IlluminaHumanMethylation450k",
                              "IlluminaHumanMethylationEPIC",
                              "IlluminaHumanMethylation27k"),
                  verbose = TRUE)
## S4 method for signature 'GenomicMethylSet'
convertArray(object,
                  outType = c("IlluminaHumanMethylation450k",
                              "IlluminaHumanMethylationEPIC",
                              "IlluminaHumanMethylation27k"),
                  verbose = TRUE)
## S4 method for signature 'GenomicRatioSet'
convertArray(object,
                  outType = c("IlluminaHumanMethylation450k",
                              "IlluminaHumanMethylationEPIC",
                              "IlluminaHumanMethylation27k"),
                  verbose = TRUE)
```

Arguments

`object`	The input object.
`outType`	The array type of the output.
`verbose`	Should the function be verbose?

Details

FIXME: describe the RGChannelSet conversion.

Value

The output object has the same class as the input object, that is either an `RGChannelSet`, a `MethylSet`, a `RatioSet`, a `GemomicMethylSet` or a `GenomicRatioSet`, with the type of the array given by the `outType` argument.

Author(s)

Jean-Philippe Fortin and Kasper D. Hansen.

Examples

```
if(require(minfiData)) {
  data(RGsetEx.sub)
  rgSet <- convertArray(RGsetEx.sub, outType = "IlluminaHumanMethylationEPIC")
```

```
  rgSet
}
```

---

cpgCollapse *Collapse methylation values of adjacent CpGs into a summary value.*

Description

This function groups adjacent loci into clusters with a specified maximum gap between CpGs in the cluster, and a specified maximum cluster width. The loci within each cluster are summarized resulting in a single methylation estimate per cluster.

Usage

```
cpgCollapse(object, what = c("Beta", "M"), maxGap = 500,
            blockMaxGap = 2.5 * 10^5, maxClusterWidth = 1500,
            dataSummary = colMeans, na.rm = FALSE,
            returnBlockInfo = TRUE, islandAnno = NULL, verbose = TRUE,
            ...)
```

Arguments

`object`	An object of class `[Genomic]MethylSet` or `[Genomic]RatioSet`.
`what`	Should operation be performed on the M-scale or Beta-scale?
`maxGap`	Maximum gap between CpGs in a cluster
`blockMaxGap`	Maximum block gap
`maxClusterWidth`	Maximum cluster width
`dataSummary`	Function used to summarize methylation across CpGs in the cluster.
`na.rm`	Should NAs be removed when summarizing? Passed on to the dataSummary function.
`returnBlockInfo`	Should the block annotation table be returned in addition to the block table?
`islandAnno`	Which Island annotation should be used. `NULL` indicates the default. This argument is only useful if the annotatio object contains more than one island annotation.
`verbose`	Should the function be verbose?
`...`	Passed on to getMethSignal and getCN. Can be used to specify

Details

This function is used as the first step of block-finding. It groups adjacent loci into clusters with a default maximum gap of 500bp and a maximum cluster width of 1,500bp. The loci within each cluster are then summarized (using the mean by default) resulting in a single methylation estimate per cluster. Cluster estimates from open-sea probes are used in block-finding.

Value

If returnBlockInfo is `FALSE`: a GenomicRatioSet of collapsed CpG clusters.

If returnBlockInfo is `TRUE`:

`object`	A GenomicRatioSet of collapsed CpG clusters
`blockInfo`	A cluster annotation data frame

Author(s)

Rafael Irizarry

See Also

`blockFinder`

`DelayedArray_utils.Rd` *Stubs for internal functions*

Description

Stubs for internal functions

`densityBeanPlot` *Density bean plots of methylation Beta values.*

Description

Density 'bean' plots of methylation Beta values, primarily for QC.

Usage

```
densityBeanPlot(dat, sampGroups = NULL, sampNames = NULL, main = NULL,
    pal = brewer.pal(8, "Dark2"), numPositions = 10000)
```

Arguments

`dat`	An `RGChannelSet`, a `MethylSet` or a `matrix`. We either use the `getBeta` function to get Beta values (for the first two) or we assume the matrix contains Beta values.
`sampGroups`	Optional sample group labels. See details.
`sampNames`	Optional sample names. See details.
`main`	Plot title.
`pal`	Color palette.
`numPositions`	The density calculation uses `numPositions` randomly selected CpG positions. If `NULL` use all positions.

Details

This function produces the density bean plot component of the QC report. If `sampGroups` is specified, group-specific colors will be used. For speed reasons the plots are produced using a random subset of CpG positions. The number of positions used is specified by the `numPositions` option.

Value

No return value. Plots are produced as a side-effect.

Author(s)

Martin Aryee <aryee@jhu.edu>.

References

P Kampstra. *Beanplot: A boxplot alternative for visual comparison of distributions.* Journal of Statistical Software 28, (2008). http://www.jstatsoft.org/v28/c01

See Also

qcReport, mdsPlot, controlStripPlot, densityPlot

Examples

```
if (require(minfiData)) {

names <- pData(RGsetEx)$Sample_Name
groups <- pData(RGsetEx)$Sample_Group
par(mar=c(5,6,4,2))
densityBeanPlot(RGsetEx, sampNames=names, sampGroups=groups)

}
```

---

densityPlot *Density plots of methylation Beta values.*

Description

Density plots of methylation Beta values, primarily for QC.

Usage

```
densityPlot(dat, sampGroups = NULL, main = "", xlab = "Beta",
    pal = brewer.pal(8, "Dark2"), xlim, ylim, add = TRUE, legend = TRUE,
    ...)
```

Arguments

`dat`	An `RGChannelSet`, a `MethylSet` or a `matrix`. We either use the `getBeta` function to get Beta values (for the first two) or we assume the matrix contains Beta values.
`sampGroups`	Optional sample group labels. See details.
`main`	Plot title.
`xlab`	x-axis label.
`pal`	Color palette.
`xlim`	x-axis limits.
`ylim`	y-axis limits.
`add`	Start a new plot?
`legend`	Plot legend.
`...`	Additional options to be passed to the plot command.

Details

This function produces the density plot component of the QC report. If `sampGroups` is specified, group-specific colors will be used.

Value

No return value. Plots are produced as a side-effect.

Author(s)

Martin Aryee <aryee@jhu.edu>.

See Also

`qcReport`, `mdsPlot`, `controlStripPlot`, `densityBeanPlot`

Examples

```
if (require(minfiData)) {

groups <- pData(RGsetEx)$Sample_Group
densityPlot(RGsetEx, sampGroups=groups)

}
```

`detectionP` *Detection p-values for all probed genomic positions.*

Description

This function identifies failed positions defined as both the methylated and unmethylated channel reporting background signal levels.

Usage

```
detectionP(rgSet, type = "m+u")
```

Arguments

`rgSet`	An `RGChannelSet`.
`type`	How to calculate p-values. Only `m+u` is currently implemented (See details).

Details

A detection p-value is returned for every genomic position in every sample. Small p-values indicate a good position. Positions with non-significant p-values (typically >0.01) should not be trusted.

The `m+u` method compares the total DNA signal (Methylated + Unmethylated) for each position to the background signal level. The background is estimated using negative control positions, assuming a normal distribution. Calculations are performed on the original (non-log) scale.

This function is different from the detection routine in Genome Studio.

Value

A matrix with detection p-values.

Author(s)

Martin Aryee <`aryee@jhu.edu`>.

Examples

```
if (require(minfiData)) {
  detP <- detectionP(RGsetEx.sub)
  failed <- detP>0.01
  colMeans(failed) # Fraction of failed positions per sample
  sum(rowMeans(failed)>0.5) # How many positions failed in >50% of samples?
}
```

dmpFinder *Find differentially methylated positions*

Description

Identify CpGs where methylation is associated with a continuous or categorical phenotype.

Usage

```
dmpFinder(dat, pheno, type = c("categorical", "continuous"),
    qCutoff = 1, shrinkVar = FALSE)
```

Arguments

`dat`	A `MethylSet` or a `matrix`.
`pheno`	The phenotype to be tested for association with methylation.
`type`	Is the phenotype '‘continuous’ or ‘categorical’?
`qCutoff`	DMPs with an FDR q-value greater than this will not be returned.
`shrinkVar`	Should variance shrinkage be used? See details.

Details

This function tests each genomic position for association between methylation and a phenotype. Continuous phenotypes are tested with linear regression, while an F-test is used for categorical phenotypes.

Variance shrinkage (`shrinkVar=TRUE`) is recommended when sample sizes are small (<10). The sample variances are squeezed by computing empirical Bayes posterior means using the **limma** package.

Value

A table with one row per CpG.

Author(s)

Martin Aryee <`aryee@jhu.edu`>.

See Also

`squeezeVar` and the **limma** package in general.

Examples

```
if (require(minfiData)) {

grp <- pData(MsetEx)$Sample_Group
MsetExSmall <- MsetEx[1:1e4,] # To speed up the example
M <- getM(MsetExSmall, type = "beta", betaThreshold = 0.001)
dmp <- dmpFinder(M, pheno=grp, type="categorical")
sum(dmp$qval < 0.05, na.rm=TRUE)
head(dmp)

}
```

`estimateCellCounts` *Cell Proportion Estimation*

Description

Estimates the relative proportion of pure cell types within a sample. For example, given peripheral blood samples, this function will return the relative proportions of lymphocytes, monocytes, B-cells, and neutrophils.

Usage

```
estimateCellCounts(rgSet, compositeCellType = "Blood",
                   processMethod = "auto", probeSelect = "auto",
                   cellTypes = c("CD8T","CD4T", "NK","Bcell","Mono","Gran"),
                   referencePlatform = c("IlluminaHumanMethylation450k",
                                         "IlluminaHumanMethylationEPIC",
                                         "IlluminaHumanMethylation27k"),
                   returnAll = FALSE, meanPlot = FALSE, verbose = TRUE, ...)
```

Arguments

`rgSet` The input `RGChannelSet` for the procedure.

`compositeCellType` Which composite cell type is being deconvoluted. Should be one of "Blood", "CordBlood", or "DLPFC". See details.

`processMethod` How should the user and reference data be processed together? Default input "auto" will use `preprocessQuantile` for Blood and DLPFC and `preprocessNoob` otherwise, in line with the existing literature. Set it to the name of a preprocessing function as a character if you want to override it, like `"preprocessFunnorm"`.

`probeSelect` How should probes be selected to distinguish cell types? Options include "both", which selects an equal number (50) of probes (with F-stat p-value < 1E-8) with the greatest magnitude of effect from the hyper- and hypo-methylated sides, and "any", which selects the 100 probes (with F-stat p-value < 1E-8) with the greatest magnitude of difference regardless of direction of effect. Default input "auto" will use "any" for cord blood and "both" otherwise, in line with previous versions of this function and/or our recommendations. Please see the references for more details.

`cellTypes` Which cell types, from the reference object, should be we use for the deconvolution? See details.

`referencePlatform` The platform for the reference dataset; if the input `rgSet` belongs to another platform, it will be converted using `convertArray`.

`returnAll` Should the composition table and the normalized user supplied data be return?

`verbose` Should the function be verbose?

`meanPlot` Whether to plots the average DNA methylation across the cell-type discrimating probes within the mixed and sorted samples.

`...` Passed to `preprocessQuantile`.

Details

This is an implementaion of the Houseman et al (2012) regression calibration approachalgorithm to the Illumina 450k microarray for deconvoluting heterogeneous tissue sources like blood. For example, this function will take an `RGChannelSet` from a DNA methylation (DNAm) study of blood, and return the relative proportions of CD4+ and CD8+ T-cells, natural killer cells, monocytes, granulocytes, and b-cells in each sample.

The function currently supports cell composition estimation for blood, cord blood, and the frontal cortex, through `compositeCellType` values of "Blood", "CordBlood", and "DLPFC", respectively. Packages containing the appropriate reference data should be installed before running the function for the first time ("FlowSorted.Blood.450k", "FlowSorted.DLPFC.450k", "FlowSorted.CordBlood.450k"). Each tissue supports the estimation of different cell types, delimited via the `cellTypes` argument. For blood, these are "Bcell", "CD4T", "CD8T", "Eos", "Gran", "Mono", "Neu", and "NK" (though the default value for `cellTypes` is often sufficient). For cord blood, these are "Bcell", "CD4T", "CD8T", "Gran", "Mono", "Neu", and "nRBC". For frontal cortex, these are "NeuN_neg" and "NeuN_pos". See documentation of individual reference packages for more details.

The `meanPlot` should be used to check for large batch effects in the data, reducing the confidence placed in the composition estimates. This plot depicts the average DNA methylation across the cell-type discrimating probes in both the provided and sorted data. The means from the provided heterogeneous samples should be within the range of the sorted samples. If the sample means fall outside the range of the sorted means, the cell type estimates will inflated to the closest cell type. Note that we quantile normalize the sorted data with the provided data to reduce these batch effects.

Value

Matrix of composition estimates across all samples and cell types.

If `returnAll=TRUE` a list of a count matrix (see previous paragraph), a composition table and the normalized user data in form of a GenomicMethylSet.

Author(s)

Andrew E. Jaffe, Shan V. Andrews, E. Andres Houseman

References

EA Houseman, WP Accomando, DC Koestler, BC Christensen, CJ Marsit, HH Nelson, JK Wiencke and KT Kelsey. *DNA methylation arrays as surrogate measures of cell mixture distribution.* BMC bioinformatics (2012) 13:86. doi:10.1186/1471-2105-13-86.

AE Jaffe and RA Irizarry. *Accounting for cellular heterogeneity is critical in epigenome-wide association studies.* Genome Biology (2014) 15:R31. doi:10.1186/gb-2014-15-2-r31.

KM Bakulski, JI Feinberg, SV Andrews, J Yang, S Brown, S McKenney, F Witter, J Walston, AP Feinberg, and MD Fallin. *DNA methylation of cord blood cell types: Applications for mixed cell birth studies.* Epigenetics (2016) 11:5. doi:10.1080/15592294.2016.1161875.

See Also

preprocessQuantile and convertArray.

Examples

```
## Not run:
if(require(FlowSorted.Blood.450k)) {
  wh.WBC <- which(FlowSorted.Blood.450k$CellType == "WBC")
  wh.PBMC <- which(FlowSorted.Blood.450k$CellType == "PBMC")
  RGset <- FlowSorted.Blood.450k[, c(wh.WBC, wh.PBMC)]
  ## The following line is purely to work around an issue with repeated
  ## sampleNames and Biobase::combine()
  sampleNames(RGset) <- paste(RGset$CellType,
    c(seq(along = wh.WBC), seq(along = wh.PBMC)), sep = "_")
  counts <- estimateCellCounts(RGset, meanPlot = FALSE)
  round(counts, 2)
}

## End(Not run)
```

---

fixMethOutliers *Fix methylation outliers*

Description

Methylation outliers (loci with very extreme values of the Meth or Unmeth channel) are identified and fixed (see details).

Usage

```
fixMethOutliers(object, K = -3, verbose = FALSE)
```

Arguments

object	An object of class [Genomic]MethylSet.
K	The number of standard deviations away from the median when defining the outlier cutoff, see details.
verbose	Should the function be verbose?

Details

This function fixes outlying methylation calls in the Meth channel and Unmeth channel separately.

Unlike other types of arrays, all loci on a methylation array ought to measure something (apart from loci on the Y chromosome in a female sample). An outlier is a loci with a very low value in one of the two methylation channels. Typically, relatively few loci ought to be outliers.

An outlier is defined in a sample and methylation channel specific way. First the (sample, methylation channel) values are `log2(x+0.5)` transformed and then the median and mad of these values are computed. An outlier is then defined to be any value less than the median plus `K` times the mad, and these outlier values are thresholded at the cutoff (on the original scale).

Value

An object of the same class as `object` where outlier values in the methylation channels have been thresholded.

Author(s)

Rafael A. Irizarry and Kasper D. Hansen

See Also

`minfiQC`

Examples

```
if(require(minfiData)) {
  MsetEx <- fixMethOutliers(MsetEx)
}
```

`gaphunter` *Find gap signals in 450k data*

Description

This function finds probes in the Illumina 450k Array for which calculated beta values cluster into distinct groups separated by a defined threshold. It identifies, for these 'gaps signals' the number of groups, the size of these groups, and the samples in each group.

Usage

```
gaphunter(object, threshold=0.05, keepOutliers=FALSE,
          outCutoff=0.01, verbose=TRUE)
```

Arguments

object	An object of class (Genomic)RatioSet, (Genomic)MethylSet, or matrix. If one of the first two, codegetBeta is used to calculate beta values. If a matrix, must be one of beta values.
threshold	The difference in consecutive, ordered beta values that defines the presence of a gap signal. Defaults to 5 percent.
keepOutliers	Should outlier-driven gap signals be kept in the results? Defaults to FALSE
outCutoff	Value used to identify gap signals driven by outliers. Defined as the percentage of the total sample size; the sum of samples in all groups except the largest must exceed this number of samples in order for the probe to still be considered a gap signal. Defaults to 1 percent.
verbose	logical value. If TRUE, it writes some messages indicating progress. If FALSE nothing should be printed.

Details

The function can calculate a beta matrix or utilize a user-supplied matrix of beta values.

The function will idenfity probes with a gap in a beta signal greater than or equal to the defined threshold. These probes constitue an additional, dataset-specific subset of probes that merit special consideration due to their tendency to be driven by an underlying SNP or other genetic variant. In this manner, these probes can serve as surrogates for underlying genetic signal locally and/or in a broader (i.e. haplotype) context. Please see our upcoming manuscript for a detailed description of the utility of these probes.

Outlier-driven gap signals are those in which the sum of the smaller group(s) does not exceed a certain percentage of the sample size, defined by the argument outCutoff.

Value

A list with three values,

proberesults	A data frame listing, for each identified gap signal, the number of groups and the size of each group.
sampleresults	a matrix of dimemsions probes (rows) by samples (columns). Individuals are assigned numbers based onthe groups into which they cluster. Lower number groups indicate lower mean methylation values for the group. For example, individuals coded as '1' will have a lower mean methylation value than those individuals coded as '2'.
algorithm	A list detailing the arguments supplied to the function.

Author(s)

Shan V. Andrews <sandre17@jhu.edu>.

References

SV Andrews, C Ladd-Acosta, AP Feinberg, KD Hansen, MD Fallin. *'Gap hunting' to characterize clustered probe signals in Illumina methylation array data.* Epigenetics & Chromatin (2016) 9:56. doi:10.1186/s13072-016-0107-z.

Examples

```
if(require(minfiData)) {
  gapres <- gaphunter(MsetEx.sub, threshold=0.3, keepOutliers=TRUE)
  #Note: the threshold argument is increased from the default value in this small example
  #dataset with 6 people to avoid the reporting of a large amount of probes as gap signals.
  #In a typical EWAS setting with hundreds of samples, the default arguments should be
  #sufficient.
}
```

GenomicMethylSet-class

GenomicMethylSet instances

Description

This class holds preprocessed data for Illumina methylation microarrays, mapped to a genomic location.

Usage

```
## Constructor

GenomicMethylSet(gr = GRanges(), Meth = new("matrix"),
                 Unmeth = new("matrix"), annotation = "",
                 preprocessMethod = "", ...)

## Data extraction / Accessors

## S4 method for signature 'GenomicMethylSet'
getMeth(object)
## S4 method for signature 'GenomicMethylSet'
getUnmeth(object)
## S4 method for signature 'GenomicMethylSet'
getBeta(object, type = "", offset = 0, betaThreshold = 0)
## S4 method for signature 'GenomicMethylSet'
getM(object, type = "", ...)
## S4 method for signature 'GenomicMethylSet'
getCN(object, ...)
## S4 method for signature 'GenomicMethylSet'
pData(object)
## S4 method for signature 'GenomicMethylSet'
sampleNames(object)
## S4 method for signature 'GenomicMethylSet'
featureNames(object)
## S4 method for signature 'GenomicMethylSet'
annotation(object)
## S4 method for signature 'GenomicMethylSet'
```

```
preprocessMethod(object)
## S4 method for signature 'GenomicMethylSet'
mapToGenome(object, ...)
```

Arguments

`object`	A `GenomicMethylSet`.
`gr`	A `GRanges` object.
`Meth`	A matrix of methylation values (between zero and infinity) with each row being a methylation loci and each column a sample.
`Unmeth`	See the `Meth` argument.
`annotation`	An annotation character string.
`preprocessMethod`	A preprocess method character string.
`type`	How are the values calculated? For `getBeta` setting `type="Illumina"` sets `offset=100` as per Genome Studio. For `getM` setting `type=""` computes M-values as the logarithm of `Meth`/`Unmeth`, otherwise it is computed as the logit of `getBeta(object)`.
`offset`	Offset in the beta ratio, see detail.
`betaThreshold`	Constrains the beta values to be in the inverval betwen `betaThreshold` and 1-`betaThreshold`.
`...`	For the constructor, additional arguments to be passed to `SummarizedExperiment`; of particular interest are `colData` and `metadata`. For `getM` these values gets passed onto `getBeta`. For `mapToGenome`, this is ignored.

Details

For a detailed discussion of `getBeta` and `getM` see the deatils section of `MethylSet`.

Value

An object of class `GenomicMethylSet` for the constructor.

Constructor

Instances are constructed using the `GenomicMethylSet` function with the arguments outlined above.

Accessors

A number of useful accessors are inherited from `RangedSummarizedExperiment`.

In the following code, `object` is a `GenomicMethylSet`.

`getMeth(object)`, `getUnmeth(object)` Get the Meth or Unmeth matrix.

`getBeta(object)` Get Beta, see details.

`getM(object)` get M-values, see details.

`getCN(object)` get copy number values which are defined as the sum of the methylation and unmethylation channel.

`getManifest(object)` get the manifest associated with the object.

`sampleNames(object)`, `featureNames(object)` Get the sampleNames (colnames) or the featureNames (rownames).

`preprocessMethod(object)`, `annotation(object)` Get the preprocess method or annotation `character`.

Utilities

`mapToGenome(object)` Since `object` is already mapped to the genome, this method simply returns `object` unchanged.

`combine`: Combines two different `GenomicMethylSet`, eventually using the `cbind` method for `SummarizedExperiment`.

Author(s)

Kasper Daniel Hansen <khansen@jhsph.edu>

See Also

`RangedSummarizedExperiment` in the **SummarizedExperiment** package for the basic class structure. Objects of this class are typically created by using the function `mapToGenome` on a `MethylSet`.

Examples

```
showClass("GenomicMethylSet")
```

GenomicRatioSet-class *GenomicRatioSet instances*

Description

This class holds preprocessed data for Illumina methylation microarrays, mapped to a genomic location.

Usage

```
## Constructor

GenomicRatioSet(gr = GRanges(), Beta = NULL, M = NULL,
                CN = NULL, annotation = "",
                preprocessMethod = "", ...)

## Data extraction / Accessors

## S4 method for signature 'GenomicRatioSet'
getBeta(object)
```

```
## S4 method for signature 'GenomicRatioSet'
getM(object)
## S4 method for signature 'GenomicRatioSet'
getCN(object)
## S4 method for signature 'GenomicRatioSet'
pData(object)
## S4 method for signature 'GenomicRatioSet'
sampleNames(object)
## S4 method for signature 'GenomicRatioSet'
featureNames(object)
## S4 method for signature 'GenomicRatioSet'
annotation(object)
## S4 method for signature 'GenomicRatioSet'
preprocessMethod(object)
## S4 method for signature 'GenomicRatioSet'
mapToGenome(object, ...)
```

Arguments

object	A GenomicRatioSet.
gr	A GRanges object.
Beta	A matrix of beta values (optional, see details).
M	A matrix of M values (optional, see details).
CN	A matrix of copy number values.
annotation	An annotation character string.
preprocessMethod	A preprocess method character string.
...	For the constructor, additional arguments to be passed to SummarizedExperiment; of particular interest are colData and metadata. For mapToGenome, this is ignored.

Details

This class holds M or Beta values (or both) together with associated genomic coordinates. It is not possible to get Meth or Unmeth values from this object. The intention is to use this kind of object as an analysis end point.

In case one of M or Beta is missing, the other is computed on the fly. For example, M is computed from Beta as the logit (base 2) of the Beta values.

Value

An object of class GenomicRatioSet for the constructor.

Constructor

Instances are constructed using the GenomicRatioSet function with the arguments outlined above.

Accessors

A number of useful accessors are inherited from `RangedSummarizedExperiment`.

In the following code, `object` is a `GenomicRatioSet`.

`getBeta(object)` Get Beta, see details.

`getM(object)` get M-values, see details.

`getCN(object)` get copy number, see details.

`getManifest(object)` get the manifest associated with the object.

`sampleNames(object)`, `featureNames(object)` Get the sampleNames (colnames) or the featureNames (rownames).

`preprocessMethod(object)`, `annotation(object)` Get the preprocess method or annotation `character`.

Utilities

`mapToGenome(object)` Since `object` is already mapped to the genome, this method simply returns `object` unchanged.

`combine`: Combines two different `GenomicRatioSet`, eventually using the `cbind` method for `SummarizedExperiment`.

Author(s)

Kasper Daniel Hansen <`khansen@jhsph.edu`>

See Also

`RangedSummarizedExperiment` in the **SummarizedExperiment** package for the basic class structure.

Examples

```
showClass("GenomicRatioSet")
```

`getAnnotation` *Accessing annotation for Illumina methylation objects*

Description

These functions access provided annotation for various Illumina methylation objects.

Usage

```
getAnnotation(object, what = "everything", lociNames = NULL,
              orderByLocation = FALSE, dropNonMapping = FALSE)

getLocations(object, mergeManifest = FALSE,
             orderByLocation = FALSE, lociNames = NULL)

getAnnotationObject(object)

getSnpInfo(object, snpAnno = NULL)
addSnpInfo(object, snpAnno = NULL)

dropLociWithSnps(object, snps = c("CpG", "SBE"), maf = 0, snpAnno = NULL)

getProbeType(object, withColor = FALSE)

getIslandStatus(object, islandAnno = NULL)
```

Arguments

object	A minfi object.
what	Which annotation objects should be returned?
lociNames	Restrict the return values to these loci.
orderByLocation	Should the return object be ordered according to genomic location.
dropNonMapping	Should loci that do not have a genomic location associated with it (by being marked as unmapped or multi) be dropped from the return object.
mergeManifest	Should the manifest be merged into the return object?
snpAnno	The snp annotation you want to use; NULL signifies picking the default.
withColor	Should the return object have the type I probe color labelled?
snps	The type of SNPs used.
maf	Minor allelle fraction.
islandAnno	Like snpAnno, but for islands.

Details

getAnnotation returns requested annotation as a DataFrame, with each row corresponding to a methylation loci. If object is of class IlluminaHumanAnnotation no specific ordering of the return object is imposed. If, on the other hand, the class of object imposes some natural order on the return object (ie. if the object is of class [Genomic](Methyl|Ratio)Set), this order is kept in the return object. Note that RGChannelSet does not impose a specific ordering on the methylation loci.

getAnnotationObject returns the annotation object, as opposed to the annotation the object contains. This is useful for printing and examining the contents of the object.

getLocations is a convenience function which returns Locations as a GRanges and which furthermore drops unmapped loci. A user should not need to call this function, instead mapToGenome should be used to get genomic coordinates and granges to return those coordinates.

getSnpInfo is a conevnience function which gets a SNP DataFrame containing information on which probes contains SNPs where. addSnpInfo adds this information to the rowRanges or granges of the object. dropLociWithSnps is a convenience function for removing loci with SNPs based on their MAF.

To see which options are available for what, simply print the annotation object, possibly using getAnnotationObject.

Value

For getAnnotation, a DataFrame with the requested information.

For getAnnotationObject, a IlluminaMethylationAnnotation object.

For getLocations, a GRanges with the locations.

For getProbeType and getIslandStatus, a character vector with the requested information.

For getSnpInfo, a DataFrame with the requested information. For addSnpInfo, an object of the same class as object but with the SNP information added to the metadata columns of the granges of the object.

For dropLociWithSnps an object of the same kind as the input, possibly with fewer loci.

Author(s)

Kasper Daniel Hansen<khansen@jhsph.edu>

See Also

IlluminaMethylationAnnotation for the basic class, mapToGenome for a better alternative (for users) to getLocations.

Examples

```
if(require(minfiData)) {
  table(getIslandStatus(MsetEx))
  getAnnotation(MsetEx, what = "Manifest")
}
```

---

getGenomicRatioSetFromGEO

Reading Illumina methylation array data from GEO.

Description

Reading Illumina methylation array data from GEO.

Usage

```
getGenomicRatioSetFromGEO(GSE = NULL, path = NULL, array = "IlluminaHumanMethylation450k",
                         annotation = .default.450k.annotation, what = c("Beta", "M"),
                                mergeManifest = FALSE, i = 1)
```

Arguments

GSE	The GSE ID of the dataset to be downloaded from GEO.
path	If data already downloaded, the path with soft files. Either GSE or path are required.
array	Array name.
annotation	The feature annotation to be used. This includes the location of features thus depends on genome build.
what	Are Beta or M values being downloaded.
mergeManifest	Should the Manifest be merged to the final object.
i	If the GEO download results in more than one dataset, it pickes entry i.

Details

This function downloads data from GEO using getGEO from the **GEOquery** package. It then returns a GenomicRatioSet object. Note that the rs probes (used for genotyping) are dropped.

Value

A GenomicRatioSet object.

Author(s)

Tim Triche Jr. and Rafael A. Irizarry<rafa@jimmy.harvard.edu>.

See Also

If the data is already in memor you can use makeGenomicRatioSetFromMatrix

Examples

```
## Not run:
mset=getGenomicRatioSetFromGEO("GSE42752")

## End(Not run)
```

getMethSignal *Various utilities*

Description

Utility functions operating on objects from the minfi package.

Usage

```
getMethSignal(object, what = c("Beta", "M"), ...)
```

Arguments

object	An object from the minfi package supporting either getBeta or getM.
what	Which signal is returned.
...	Passed to the method described by argument what.

Value

A matrix.

Author(s)

Kasper Daniel Hansen <khansen@jhsph.edu>.

Examples

```
if(require(minfiData)) {
  head(getMethSignal(MsetEx, what = "Beta"))
}
```

getQC *Estimate sample-specific quality control (QC) for methylation data*

Description

Estimate sample-specific quality control (QC) for methylation data.

Usage

```
getQC(object)
addQC(object, qc)
plotQC(qc, badSampleCutoff = 10.5)
```

Arguments

- `object` — An object of class `[Genomic]MethylSet`.
- `qc` — An object as produced by `getQC`.
- `badSampleCutoff` — The cutoff for identifying a bad sample.

Value

For `getQC`, a `DataFrame` with two columns: `mMed` and `uMed` which are the chipwide medians of the Meth and Unmeth channels.

For `addQC`, essentially `object` supplied to the function, but with two new columns added to the pheno data slot: `uMed` and `mMed`.

Author(s)

Rafael A. Irizarry and Kasper D. Hansen

See Also

`minfiQC` for an all-in-one function.

Examples

```
if(require(minfiData)){
  qc <- getQC(MsetEx)
  MsetEx <- addQC(MsetEx, qc = qc)
  ## plotQC(qc)
}
```

---

getSex — *Estimating sample sex based on methylation data*

Description

Estimates samples sex based on methylation data.

Usage

```
getSex(object = NULL, cutoff = -2)
addSex(object, sex = NULL)
plotSex(object, id = NULL)
```

Arguments

object	An object of class [Genomic]MethylSet.
cutoff	What should the difference in log2 copynumber be between males and females.
sex	An optional character vector of sex (with values M and F).
id	Text used as plotting symbols in the plotSex function. Used for sample identification on the plot.

Details

Estimation of sex is based on the median values of measurements on the X and Y chromosomes respectively. If yMed - xMed is less than cutoff we predict a female, otherwise male.

Value

For getSex, a DataFrame with columns predictedSex (a character with values M and F), xMed and yMed, which are the chip-wide medians of measurements on the two sex chromosomes.

For addSex, an object of the same type as object but with the output of getSex(object) added to the pheno data.

For plotSex, a plot of xMed vs. yMed, which are the chip-wide medians of measurements on the two sex chromosomes, coloured by predictedSex.

Author(s)

Rafael A. Irizarry, Kasper D. Hansen, Peter F. Hickey

Examples

```
if(require(minfiData)) {
  GMsetEx <- mapToGenome(MsetEx)
  estSex <- getSex(GMsetEx)
  GMsetEx <- addSex(GMsetEx, sex = estSex)
}
```

IlluminaMethylationAnnotation-class

Class IlluminaMethylationAnnotation

Description

This is a class for representing annotation associated with an Illumina methylation microarray. Annotation is transient in the sense that it may change over time, wheres the information stored in the IlluminaMethylationManifest class only depends on the array design.

Usage

```
## Constructor
IlluminaMethylationAnnotation(objectNames, annotation = "",
                              defaults = "", packageName = "")

## Data extraction
## S4 method for signature 'IlluminaMethylationAnnotation'
getManifest(object)
```

Arguments

`object`	An object of class `IlluminaMethylationAnnotation`.
`annotation`	An annotation `character`.
`defaults`	A vector of default choices for `getAnnotation(what = "everything")`.
`objectNames`	a `character` with object names used in the package.
`packageName`	The name of the package this object will be contained in.

Value

An object of class `IlluminaMethylationAnnotation`.

Utilitues

In the following code, `object` is a `IlluminaMethylationAnnotation`.

`getManifest(object)` Get the manifest object associated with the array.

Author(s)

Kasper Daniel Hansen <`khansen@jhsph.edu`>.

See Also

`IlluminaMethylationManifest`

`IlluminaMethylationManifest-class`
Class `"IlluminaMethylationManifest"`

Description

This is a class for representing an Illumina methylation microarray design, ie. the physical location and the probe sequences. This information should be independent of genome build and annotation.

Usage

```
## Constructor

IlluminaMethylationManifest(TypeI = DataFrame(),
                            TypeII = DataFrame(),
                            TypeControl = DataFrame(),
                            TypeSnpI = DataFrame(),
                            TypeSnpII = DataFrame(),
                            annotation = "")

## Data extraction
## S4 method for signature 'IlluminaMethylationManifest'
getManifest(object)
## S4 method for signature 'character'
getManifest(object)
getProbeInfo(object, type = c("I", "II", "Control",
                              "I-Green", "I-Red", "SnpI", "SnpII"))
getManifestInfo(object, type = c("nLoci", "locusNames"))
getControlAddress(object, controlType = c("NORM_A", "NORM_C",
                                          "NORM_G", "NORM_T"),
                  asList = FALSE)
```

Arguments

object	Either an object of class IlluminaMethylationManifest or class character for getManifest. For getProbeInfo, getManifestInfo and getControlAddress an object of either class RGChannelSet, IlluminaMethylationManifest.
TypeI	A DataFrame of type I probes.
TypeII	A DataFrame of type II probes.
TypeControl	A DataFrame of control probes.
TypeSnpI	A DataFrame of SNP type I probes.
TypeSnpII	A DataFrame of SNP type II probes.
annotation	An annotation character.
type	A single character describing what kind of information should be returned. For getProbeInfo it represents the following subtypes of probes on the array: Type I, Type II, Controls as well as Type I (methylation measured in the Green channel) and Type II (methylation measured in the Red channel). For getManifestInfo it represents either the number of methylation loci (approx. number of CpGs) on the array or the locus names.
controlType	A character vector of control types.
asList	If TRUE the return object is a list with one component for each controlType.

Value

An object of class IlluminaMethylationManifest for the constructor.

Details

The data slot contains the following objects: TypeI, TypeII and TypeControl which are all of class data.frame, describing the array design.

Methylation loci of type I are measured using two different probes, in either the red or the green channel. The columns AddressA, AddresB describes the physical location of the two probes on the array (with ProbeSeqA, ProbeSeqB giving the probe sequences), and the column Color describes which color channel is used.

Methylation loci of type II are measured using a single probe, but with two different color channels. The methylation signal is always measured in the green channel.

Utilities

In the following code, object is a IlluminaMethylationManifest.

getManifest(object) Get the manifest object.

getProbeInfo(object) Returns a DataFrame giving the type I, type II or control probes. It is also possible to get the type I probes measured in the Green or Red channel. This function ensures that the return object only contains probes which are part of the input object. In case of a RGChannelSet and type I probes, both addresses needs to be in the object.

getManifestInfo(object) Get some information about the manifest object (the chip design).

getControlAddress(object) Get the control addresses for control probes of a certain type.

getControlTypes(object) Returns the types and the numbers of control probes of each type.

Author(s)

Kasper Daniel Hansen <khansen@jhsph.edu>.

See Also

IlluminaMethylationAnnotation for annotation information for the array (information depending on a specific genome build).

Examples

```
if(require(IlluminaHumanMethylation450kmanifest)) {

show(IlluminaHumanMethylation450kmanifest)
head(getProbeInfo(IlluminaHumanMethylation450kmanifest, type = "I"))
head(IlluminaHumanMethylation450kmanifest@data$TypeI)
head(IlluminaHumanMethylation450kmanifest@data$TypeII)
head(IlluminaHumanMethylation450kmanifest@data$TypeControl)

}
```

`logit2` *logit in base 2.*

Description

Utility functions for computing logit and inverse logit in base 2.

Usage

```
logit2(x)
ilogit2(x)
```

Arguments

`x`	A numeric vector.

Value

A numeric vector.

Author(s)

Kasper Daniel Hansen <khansen@jhsph.edu>.

Examples

```
logit2(c(0.25, 0.5, 0.75))
```

`makeGenomicRatioSetFromMatrix`
Make a GenomicRatioSet from a matrix

Description

Make a GenomicRatioSet from a matrix.

Usage

```
makeGenomicRatioSetFromMatrix(mat, rownames = NULL, pData = NULL,
                              array = "IlluminaHumanMethylation450k",
                              annotation = .default.450k.annotation,
                              mergeManifest = FALSE, what = c("Beta", "M"))
```

Arguments

mat	The matrix that will be converted.
rownames	The feature IDs associated with the rows of mat that will be used to match to the IlluminaHumanMethylation450k feature IDs.
pData	A DataFrame or data.frame describing the samples represented by the columns of mat. If the rownames of the pData don't match the colnames of mat these colnames will be changed. If pData is not supplied, a minimal DataFrame is created.
array	Array name.
annotation	The feature annotation to be used. This includes the location of features thus depends on genome build.
mergeManifest	Should the Manifest be merged to the final object.
what	Are Beta or M values being downloaded.

Details

Many 450K data is provided as csv files. This function permits you to convert a matrix of values into the class that is used by functions such as bumphunter and blockFinder. The rownames of mat are used to match the 450K array features. Alternatively the rownames can be supplied directly through rownames.

Value

A GenomicRatioSet object.

Author(s)

Rafael A. Irizarry<rafa@jimmy.harvard.edu>.

See Also

getGenomicRatioSetFromGEO is similar but reads data from GEO.

Examples

```
mat <- matrix(10,5,2)
rownames(mat) <- c( "cg13869341", "cg14008030","cg12045430", "cg20826792","cg00381604")
grset <- makeGenomicRatioSetFromMatrix(mat)
```

mapToGenome-methods *Mapping methylation data to the genome*

Description

Mapping Ilumina methylation array data to the genome using an annotation package. Depending on the genome, not all methylation loci may have a genomic position.

Usage

```
## S4 method for signature 'MethylSet'
mapToGenome(object, mergeManifest = FALSE)
## S4 method for signature 'MethylSet'
mapToGenome(object, mergeManifest = FALSE)
## S4 method for signature 'RGChannelSet'
mapToGenome(object, ...)
```

Arguments

object	Either a MethylSet, a RGChannelSet or a RatioSet.
mergeManifest	Should the information in the associated manifest package be merged into the location GRanges?
...	Passed to the method for MethylSet.

Details

FIXME: details on the MethylSet method.

The RGChannelSet method of this function is a convenience function: the RGChannelSet is first transformed into a MethylSet using preprocessRaw. The resulting MethylSet is then mapped directly to the genome.

This function silently drops loci which cannot be mapped to a genomic position, based on the associated annotation package.

Value

An object of class GenomicMethylSet or GenomicRatioSet.

Author(s)

Kasper Daniel Hansen <khansen@jhsph.edu>

See Also

GenomicMethylSet for the output object and MethylSet for the input object. Also, getLocations obtains the genomic locations for a given object.

Examples

```
if (require(minfiData)) {
  ## MsetEx.sub is a small subset of MsetEx;
  ## only used for computational speed.
  GMsetEx.sub <- mapToGenome(MsetEx.sub)
}
```

`mdsPlot` *Multi-dimensional scaling plots giving an overview of similarities and differences between samples.*

Description

Multi-dimensional scaling (MDS) plots showing a 2-d projection of distances between samples.

Usage

```
mdsPlot(dat, numPositions = 1000, sampNames = NULL, sampGroups = NULL, xlim, ylim,
    pch = 1, pal = brewer.pal(8, "Dark2"), legendPos = "bottomleft",
    legendNCol, main = NULL)
```

Arguments

`dat`	An `RGChannelSet`, a `MethylSet` or a `matrix`. We either use the `getBeta` function to get Beta values (for the first two) or we assume the matrix contains Beta values.
`numPositions`	Use the `numPositions` genomic positions with the most methylation variability when calculating distance between samples.
`sampNames`	Optional sample names. See details.
`sampGroups`	Optional sample group labels. See details.
`xlim`	x-axis limits.
`ylim`	y-axis limits.
`pch`	Point type. See `par` for details.
`pal`	Color palette.
`legendPos`	The legend position. See `legend` for details.
`legendNCol`	The number of columns in the legend. See `legend` for details.
`main`	Plot title.

Details

Euclidean distance is calculated between samples using the `numPositions` most variable CpG positions. These distances are then projected into a 2-d plane using classical multidimensional scaling transformation.

Value

No return value. Plots are produced as a side-effect.

Author(s)

Martin Aryee <aryee@jhu.edu>.

References

I Borg, P Groenen. *Modern Multidimensional Scaling: theory and applications (2nd ed.)* New York: Springer-Verlag (2005) pp. 207-212. ISBN 0387948457.

http://en.wikipedia.org/wiki/Multidimensional_scaling

See Also

qcReport, controlStripPlot, densityPlot, densityBeanPlot, par, legend

Examples

```
if (require(minfiData)) {

names <- pData(MsetEx)$Sample_Name
groups <- pData(MsetEx)$Sample_Group
mdsPlot(MsetEx, sampNames=names, sampGroups=groups)

}
```

MethylSet-class *MethylSet instances*

Description

This class holds preprocessed data for Illumina methylation microarrays.

Usage

```
## Constructor

MethylSet(Meth = new("matrix"), Unmeth = new("matrix"),
          annotation = "", preprocessMethod = "", ...)

## Data extraction / Accessors

## S4 method for signature 'MethylSet'
getMeth(object)
## S4 method for signature 'MethylSet'
getUnmeth(object)
```

```
## S4 method for signature 'MethylSet'
getBeta(object, type = "", offset = 0, betaThreshold = 0)
## S4 method for signature 'MethylSet'
getM(object, type = "", ...)
## S4 method for signature 'MethylSet'
getCN(object, ...)
## S4 method for signature 'MethylSet'
getManifest(object)
## S4 method for signature 'MethylSet'
preprocessMethod(object)
## S4 method for signature 'MethylSet'
annotation(object)
## S4 method for signature 'MethylSet'
pData(object)
## S4 method for signature 'MethylSet'
sampleNames(object)
## S4 method for signature 'MethylSet'
featureNames(object)

## Utilities
dropMethylationLoci(object, dropRS = TRUE, dropCH = TRUE)
```

Arguments

`object`	A `MethylSet`.
`Meth`	A matrix of methylation values (between zero and infinity) with each row being a methylation loci and each column a sample.
`Unmeth`	See the `Meth` argument.
`annotation`	An annotation string, optional.
`preprocessMethod`	A `character`, optional.
`type`	How are the values calculated? For `getBeta` setting `type="Illumina"` sets `offset=100` as per Genome Studio. For `getM` setting `type=""` computes M-values as the logarithm of `Meth`/`Unmeth`, otherwise it is computed as the logit of `getBeta(object)`.
`offset`	Offset in the beta ratio, see detail.
`betaThreshold`	Constrains the beta values to be in the inverval betwen `betaThreshold` and 1-`betaThreshold`.
`dropRS`	Should SNP probes be dropped?
`dropCH`	Should CH probes be dropped
`...`	For the constructor, additional arguments to be passed to `SummarizedExperiment`; of particular interest are `colData`, `rowData` and `metadata`. For `getM` these values gets passed onto `getBeta`.

Details

This class inherits from `eSet`. Essentially the class is a representation of a `Meth` matrix and a `Unmeth` matrix linked to a `pData` data frame.

In addition, an annotation and a preprocessMethod slot is present. The annotation slot describes the type of array and also which annotation package to use. The preprocessMethod slot describes the kind of preprocessing that resulted in this dataset.

A `MethylSet` stores `meth` and `Unmeth`. From these it is easy to compute Beta values, defined as

$$\beta = \frac{\text{Meth}}{\text{Meth} + \text{Unmeth} + \text{offset}}$$

The offset is chosen to avoid dividing with small values. Illumina uses a default of 100. M-values (an unfortunate bad name) are defined as

$$M = \text{logit}(\beta) = \log(\text{Meth}/\text{Unmeth})$$

This formula has problems if either Meth or Unmeth is zero. For this reason, we can use `betaThreshold` to make sure Beta is neither 0 nor 1, before taken the logit. What makes sense for the `offset` and `betaThreshold` depends crucially on how the data was preprocessed. Do not expect the default values to be particular good.

Value

An object of class `MethylSet` for the constructor.

Constructor

Instances are constructed using the `MethylSet` function with the arguments outlined above.

Accessors

In the following code, `object` is a `MethylSet`.

`getMeth(object)`, `getUnmeth(object)` Get the Meth or the Unmeth matrix

`getBeta(object)` Get Beta, see details.

`getM(object)` get M-values, see details.

`getCN(object)` get copy number values which are defined as the sum of the methylation and unmethylation channel.

`getManifest(object)` get the manifest associated with the object.

`preprocessMethod(object)` Get the preprocess method `character`.

Utilities

In the following code, `object` is a `MethylSet`.

`dropMethylationLoci`**(object)** A unifed interface to removing methylation loci. You can drop SNP probes (probes that measure SNPs, not probes containing SNPs) or CH probes (non-CpG methylation).

`combine`**:** Combines two different `MethylSet`, eventually using the `combine` method for `eSet`.

Author(s)

Kasper Daniel Hansen <khansen@jhsph.edu>

See Also

eSet for the basic class structure. Objects of this class are typically created from an RGChannelSet using preprocessRaw or another preprocessing function.

Examples

```
showClass("MethylSet")
```

---

minfi-defunct *Defunct functions in package 'minfi'*

Description

These functions are provided now defunct in 'minfi'.

Details

The following functions are now defunct (not working anymore); use the replacement indicated below:

- read.450k: Use read.metharray
- read.450k.sheet: Use read.metharray.sheet
- read.450k.exp: Use read.metharray.exp

See Also

Defunct.

---

minfi-deprecated *Deprecated functions in package 'minfi'*

Description

These functions are provided for compatibility with older versions of 'minfi' only, and will be defunct at the next release.

Details

No functions are currently deprecated.

The following functions are deprecated and will be made defunct; use the replacement indicated below:

- `read.450k`: `read.metharray`
- `read.450k.sheet`: `read.metharray.sheet`
- `read.450k.exp`: `read.metharray.exp`

`minfiQC` *easy one-step QC of methylation object*

Description

This function combines a number of functions into a simple to use, one step QC step/

Usage

```
minfiQC(object, fixOutliers = TRUE, verbose = FALSE)
```

Arguments

`object`	An object of class `[Genomic]MethylSet`.
`fixOutliers`	Should the function fix outlying observations (using `fixMethOutliers`) before running QC?
`verbose`	Should the function be verbose?

Details

A number of functions are run sequentially on the `object`.

First outlier values are thresholded using `fixMethOutliers`. Then qc is performed using `getQC` and then sample specific sex is estimated using `getSex`.

Value

A list with two values,

`object`	The `object` processed by `fixMethOutliers` and with a column `predictedSex` added to the pheno data.
`qc`	A `DataFrame` with columns from the output of `getQC` and `getSex`

Author(s)

Kasper D. Hansen

See Also

getSex, getQC, fixMethOutliers

Examples

```
if(require(minfiData)) {
  out <- minfiQC(MsetEx)
  ## plotQC(out$qc)
  ## plotSex(out$sex)
}
```

---

plotBetasByType — *Plot the overall distribution of beta values and the distributions of the Infinium I and II probe types.*

Description

Plot the overall density distribution of beta values and the density distributions of the Infinium I and II probe types.

Usage

```
plotBetasByType(data, probeTypes = NULL, legendPos = "top",
                colors = c("black", "red", "blue"),
                main = "", lwd = 3, cex.legend = 1)
```

Arguments

data	A MethylSet or a matrix or a vector. We either use the getBeta function to get Beta values (in the first case) or we assume the matrix or vector contains Beta values.
probeTypes	If data is a MethylSet this argument is not needed. Otherwise, a data.frame with a column 'Name' containing probe IDs and a column 'Type' containing their corresponding assay design type.
legendPos	The x and y co-ordinates to be used to position the legend. They can be specified by keyword or in any way which is accepted by xy.coords. See legend for details.
colors	Colors to be used for the different beta value density distributions. Must be a vector of length 3.
main	Plot title.
lwd	The line width to be used for the different beta value density distributions.
cex.legend	The character expansion factor for the legend text.

Details

The density distribution of the beta values for a single sample is plotted. The density distributions of the Infinium I and II probes are then plotted individually, showing how they contribute to the overall distribution. This is useful for visualising how using `preprocessSWAN` affects the data.

Value

No return value. Plot is produced as a side-effect.

Author(s)

Jovana Maksimovic `<jovana.maksimovic@mcri.edu.au>`.

See Also

`densityPlot`, `densityBeanPlot`, `par`, `legend`

Examples

```
## Not run:
if (require(minfiData)) {
  Mset.swan <- preprocessSWAN(RGsetEx, MsetEx)
  par(mfrow=c(1,2))
  plotBetasByType(MsetEx[,1], main="Raw")
  plotBetasByType(Mset.swan[,1], main="SWAN")
}

## End(Not run)
```

`plotCpg` *Plot methylation values at an single genomic position*

Description

Plot single-position (single CpG) methylation values as a function of a categorical or continuous phenotype

Usage

```
plotCpg(dat, cpg, pheno, type = c("categorical", "continuous"),
    measure = c("beta", "M"), ylim = NULL, ylab = NULL, xlab = "",
    fitLine = TRUE, mainPrefix = NULL, mainSuffix = NULL)
```

Arguments

dat	An RGChannelSet, a MethylSet or a matrix. We either use the getBeta (or getM for measure="M") function to get Beta values (or M-values) (for the first two) or we assume the matrix contains Beta values (or M-values).
cpg	A character vector of the CpG position identifiers to be plotted.
pheno	A vector of phenotype values.
type	Is the phenotype categorical or continuous?
measure	Should Beta values or log-ratios (M) be plotted?
ylim	y-axis limits.
ylab	y-axis label.
xlab	x-axis label.
fitLine	Fit a least-squares best fit line when using a continuous phenotype.
mainPrefix	Text to prepend to the CpG name in the plot main title.
mainSuffix	Text to append to the CpG name in the plot main title.

Details

This function plots methylation values (Betas or log-ratios) at individual CpG loci as a function of a phenotype.

Value

No return value. Plots are produced as a side-effect.

Author(s)

Martin Aryee <aryee@jhu.edu>.

Examples

```
if (require(minfiData)) {

grp <- pData(MsetEx)$Sample_Group
cpgs <- c("cg00050873", "cg00212031", "cg26684946", "cg00128718")
par(mfrow=c(2,2))
plotCpg(MsetEx, cpg=cpgs, pheno=grp, type="categorical")

}
```

`preprocessFunnorm` *Functional normalization for Illumina 450k arrays*

Description

Functional normalization (FunNorm) is a between-array normalization method for the Illumina Infinium HumanMethylation450 platform. It removes unwanted variation by regressing out variability explained by the control probes present on the array.

Usage

```
preprocessFunnorm(rgSet, nPCs=2, sex = NULL, bgCorr = TRUE,
                  dyeCorr = TRUE, keepCN = TRUE, ratioConvert = TRUE,
                  verbose = TRUE)
```

Arguments

`rgSet`	An object of class `RGChannelSet`.
`nPCs`	Number of principal components from the control probes PCA
`sex`	An optional numeric vector containing the sex of the samples.
`bgCorr`	Should the NOOB background correction be done, prior to functional normalization (see `preprocessNoob`)
`dyeCorr`	Should dye normalization be done as part of the NOOB background correction (see `preprocessNoob`)?
`keepCN`	Should copy number estimates be kept around? Setting to `FALSE` will decrease the size of the output object significantly.
`ratioConvert`	Should we run `ratioConvert`, ie. should the output be a `GenomicRatioSet` or should it be kept as a `GenomicMethylSet`; the latter is for experts.
`verbose`	Should the function be verbose?

Details

This function implements functional normalization preprocessing for Illumina methylation microarrays. Functional normalization extends the idea of quantile normalization by adjusting for known covariates measuring unwanted variation. For the 450k array, the first k principal components of the internal control probes matrix play the role of the covariates adjusting for technical variation. The number k of principal components can be set by the argument `nPCs`. By default `nPCs` is set to 2, and have been shown to perform consistently well across different datasets. This parameter should only be modified by expert users. The normalization procedure is applied to the Meth and Unmeth intensities separately, and to type I and type II signals separately. For the probes on the X and Y chromosomes we normalize males and females separately using the gender information provided in the `sex` argument. For the Y chromosome, standard quantile normalization is used due to the small number of probes, which results in instability for functional normalization. If sex is unspecified (`NULL`), a guess is made using by the `getSex` function using copy number information. Note that this algorithm does not rely on any assumption and therefore can be be applicable for cases where global changes are expected such as in cancer-normal comparisons or tissue differences.

Value

an object of class GenomicRatioSet, unless ratioConvert=FALSE in which case an object of class GenomicMethylSet.

Author(s)

Jean-Philippe Fortin <jfortin@jhsph.edu>, Kasper D. Hansen <khansen@jhsph.edu>.

References

JP Fortin, A Labbe, M Lemire, BW Zanke, TJ Hudson, EJ Fertig, CMT Greenwood and KD Hansen. *Functional normalization of 450k methylation array data improves replication in large cancer studies*. (2014) Genome Biology (2014) 15:503. doi:10.1186/s13059-014-0503-2.

See Also

RGChannelSet as well as IlluminaMethylationManifest for the basic classes involved in these functions. preprocessRaw and preprocessQuantile are other preprocessing functions. Background correction may be done using preprocessNoob.

Examples

```
if (require(minfiData)) {
  ## RGsetEx.sub is a small subset of RGsetEx;
  ## only used for computational speed.
  Mset.sub.funnorm <- preprocessFunnorm(RGsetEx.sub)
}
```

---

preprocessIllumina *Perform preprocessing as Genome Studio.*

Description

These functions implements preprocessing for Illumina methylation microarrays as used in Genome Studio, the standard software provided by Illumina.

Usage

```
preprocessIllumina(rgSet, bg.correct = TRUE, normalize = c("controls", "no"),
    reference = 1)
bgcorrect.illumina(rgSet)
normalize.illumina.control(rgSet, reference = 1)
```

Arguments

rgSet	An object of class RGChannelSet.
bg.correct	logical, should background correction be performed?
normalize	logical, should (control) normalization be performed?
reference	for control normalization, which array is the reference?

Details

We have reverse engineered the preprocessing methods from Genome Studio, based on the documentation.

The current implementation of control normalization is equal to what Genome Studio provides (this statement is based on comparing Genome Studio output to the output of this function), with the following caveat: this kind of normalization requires the selection of a reference array. It is unclear how Genome Studio selects the reference array, but we allow for the manual specification of this parameter.

The current implementation of background correction is roughly equal to Genome Studio. Based on examining the output of 24 arrays, we are able to exactly recreate 18 out of the 24. The remaining 6 arrays had a max discrepancy in the Red and/or Green channel of 1-4 (this is on the unlogged intensity scale, so 4 is very small).

A script for doing this comparison may be found in the `scripts` directory (although it is of limited use without the data files).

Value

`preprocessIllumina` returns a `MethylSet`, while `bgcorrect.illumina` and `normalize.illumina.control` both return a `RGChannelSet` with corrected color channels.

Author(s)

Kasper Daniel Hansen <khansen@jhsph.edu>.

See Also

`RGChannelSet` and `MethylSet` as well as `IlluminaMethylationManifest` for the basic classes involved in these functions. `preprocessRaw` is another basic preprocessing function.

Examples

```
if (require(minfiData)) {

dat <- preprocessIllumina(RGsetEx, bg.correct=FALSE, normalize="controls")
slot(name="preprocessMethod", dat)[1]

}
```

`preprocessNoob` *The Noob/ssNoob preprocessing method for Infinium methylation microarrays.*

Description

Noob (normal-exponential out-of-band) is a background correction method with dye-bias normalization for Illumina Infinium methylation arrays.

Usage

```
preprocessNoob(rgSet, offset = 15, dyeCorr = TRUE, verbose = FALSE,
                dyeMethod=c("single", "reference"))
```

Arguments

rgSet	An object of class RGChannelSet.
offset	An offset for the normexp background correction.
dyeCorr	Should dye correction be done?
verbose	Should the function be verbose?
dyeMethod	How should dye bias correction be done: use a single sample approach (ss-Noob), or a reference array?

Value

An object of class MethylSet.

Author(s)

Tim Triche, Jr.

References

TJ Triche, DJ Weisenberger, D Van Den Berg, PW Laird and KD Siegmund *Low-level processing of Illumina Infinium DNA Methylation BeadArrays*. Nucleic Acids Res (2013) 41, e90. doi:10.1093/nar/gkt090.

See Also

RGChannelSet as well as IlluminaMethylationManifest for the basic classes involved in these functions. preprocessRaw and preprocessQuantile are other preprocessing functions.

Examples

```
if (require(minfiData)) {
  ## RGsetEx.sub is a small subset of RGsetEx;
  ## only used for computational speed.
  MsetEx.sub.noob <- preprocessNoob(RGsetEx.sub)
}
## Not run:
if (require(minfiData)) {
  dyeMethods <- c(ssNoob="single", refNoob="reference")
  GRsets <- lapply(dyeMethods,
                   function(m) preprocessNoob(RGsetEx, dyeMethod=m))
  all.equal(getBeta(GRsets$refNoob), getBeta(GRsets$ssNoob)) # TRUE
}

## End(Not run)
```

preprocessQuantile *Stratified quantile normalization for an Illumina methylation array.*

Description

Stratified quantile normalization for Illumina amethylation arrays.

This function implements stratified quantile normalization preprocessing for Illumina methylation microarrays. Probes are stratified by region (CpG island, shore, etc.)

Usage

```
preprocessQuantile(object, fixOutliers = TRUE, removeBadSamples = FALSE,
                   badSampleCutoff = 10.5, quantileNormalize = TRUE,
                   stratified = TRUE, mergeManifest = FALSE, sex = NULL,
                   verbose = TRUE)
```

Arguments

`object`	An object of class `RGChannelSet` or `[Genomic]MethylSet`.
`fixOutliers`	Should low outlier Meth and Unmeth signals be fixed?
`removeBadSamples`	Should bad samples be removed?
`badSampleCutoff`	Samples with median Meth and Umneth signals below this cutoff will be labelled 'bad'.
`quantileNormalize`	Should quantile normalization be performed?
`stratified`	Should quantile normalization be performed within genomic region strata (e.g. CpG island, shore, etc.)?
`mergeManifest`	Should the information in the associated manifest package be merged into the output object?
`sex`	Gender
`verbose`	Should the function be verbose?

Details

This function implements stratified quantile normalization preprocessing for Illumina methylation microarrays. If `removeBadSamples` is `TRUE` we calculate the median Meth and median Unmeth signal for each sample, and remove those samples where their average falls below `badSampleCutoff`. The normalization procedure is applied to the Meth and Unmeth intensities separately. The distribution of type I and type II signals is forced to be the same by first quantile normalizing the type II probes across samples and then interpolating a reference distribution to which we normalize the type I probes. Since probe types and probe regions are confounded and we know that DNAm distributions vary across regions we stratify the probes by region before applying this interpolation. For the probes on the X and Y chromosomes we normalize males and females separately using the gender information provided in the `sex` argument. If gender is unspecified (`NULL`), a guess is made using by the `getSex` function using copy number information. Background correction is not used, but very small intensities close to zero are thresholded using the `fixMethOutlier`. Note that this algorithm relies on the assumptions necessary for quantile normalization to be applicable and thus is not recommended for cases where global changes are expected such as in cancer-normal comparisons.

Note that this normalization procedure is essentially similar to one previously presented (Touleimat and Tost, 2012), but has been independently re-implemented due to the present lack of a released, supported version.

Value

a `GenomicRatioSet`

Note

A bug in the function was found to affect the Beta values of type I probes, when `stratified=TRUE` (default). This is fixed in minfi version 1.19.7 and 1.18.4 and greater.

Author(s)

Rafael A. Irizarry

References

N Touleimat and J Tost. *Complete pipeline for Infinium Human Methylation 450K BeadChip data processing using subset quantile normalization for accurate DNA methylation estimation.* Epigenomics (2012) 4:325-341.

See Also

`getSex`, `minfiQC`, `fixMethOutliers` for functions used as part of `preprocessQuantile`.

Examples

```
if (require(minfiData)) {
  # NOTE: RGsetEx.sub is a small subset of RGsetEx; only used for computational
  #       speed
  GMset.sub.quantile <- preprocessQuantile(RGsetEx.sub)
}
## Not run:
if(require(minfiData)) {
  GMset <- preprocessQuantile(RGsetEx)
}

## End(Not run)
```

preprocessRaw *Creation of a MethylSet without normalization*

Description

Converts the Red/Green channel for an Illumina methylation array into methylation signal, without using any normalization.

Usage

```
preprocessRaw(rgSet)
```

Arguments

rgSet	An object of class RGChannelSet.

Details

This function takes the Red and the Green channel of an Illumina methylation array, together with its associated manifest object and converts it into a MethylSet containing the methylated and unmethylated signal.

Value

An object of class MethylSet

Author(s)

Kasper Daniel Hansen<khansen@jhsph.edu>.

See Also

RGChannelSet and MethylSet as well as IlluminaMethylationManifest.

Examples

```
if (require(minfiData)) {

dat <- preprocessRaw(RGsetEx)
slot(name="preprocessMethod", dat)[1]

}
```

preprocessSWAN *Subset-quantile Within Array Normalisation for Illumina Infinium HumanMethylation450 BeadChips*

Description

Subset-quantile Within Array Normalisation (SWAN) is a within array normalisation method for the Illumina Infinium HumanMethylation450 platform. It allows Infinium I and II type probes on a single array to be normalized together.

Usage

```
preprocessSWAN(rgSet, mSet = NULL, verbose = FALSE)
```

Arguments

`rgSet`	An object of class `RGChannelSet`.
`mSet`	An optional object of class `MethylSet`. If set to `NULL` preprocessSwan uses `preprocessRaw` on the `rgSet` argument. In case `mSet` is supplied, make sure it is the result of preprocessing the `rgSet` argument.
`verbose`	Should the function be verbose?

Details

The SWAN method has two parts. First, an average quantile distribution is created using a subset of probes defined to be biologically similar based on the number of CpGs underlying the probe body. This is achieved by randomly selecting N Infinium I and II probes that have 1, 2 and 3 underlying CpGs, where N is the minimum number of probes in the 6 sets of Infinium I and II probes with 1, 2 or 3 probe body CpGs. If no probes have previously been filtered out e.g. sex chromosome probes, etc. N=11,303. This results in a pool of 3N Infinium I and 3N Infinium II probes. The subset for each probe type is then sorted by increasing intensity. The value of each of the 3N pairs of observations is subsequently assigned to be the mean intensity of the two probe types for that row or "quantile". This is the standard quantile procedure. The intensities of the remaining probes are then separately adjusted for each probe type using linear interpolation between the subset probes.

Value

an object of class `MethylSet`

Note

SWAN uses a random subset of probes to do the between array normalization. In order to achive reproducible results, the seed needs to be set using `set.seed`.

Author(s)

Jovana Maksimovic<`jovana.maksimovic@mcri.edu.au`>

References

J Maksimovic, L Gordon and A Oshlack (2012). *SWAN: Subset quantile Within-Array Normalization for Illumina Infinium HumanMethylation450 BeadChips*. Genome Biology 13, R44.

See Also

RGChannelSet and MethylSet as well as IlluminaMethylationManifest.

Examples

```
if (require(minfiData)) {
  ## RGsetEx.sub is a small subset of RGsetEx;
  ## only used for computational speed.
  MsetEx.sub.swan <- preprocessSWAN(RGsetEx.sub)
}
## Not run:
if (require(minfiData)) {
  dat <- preprocessRaw(RGsetEx)
  preprocessMethod(dat)
  datSwan <- preprocessSWAN(RGsetEx, mSet = dat)
  datIlmn <- preprocessIllumina(RGsetEx)
  preprocessMethod(datIlmn)
  datIlmnSwan <- preprocessSWAN(RGsetEx, mSet = datIlmn)
}

## End(Not run)
```

---

qcReport — *QC report for Illumina Infinium Human Methylation 450k arrays*

Description

Produces a PDF QC report for Illumina Infinium Human Methylation 450k arrays, useful for identifying failed samples.

Usage

```
qcReport(rgSet, sampNames = NULL, sampGroups = NULL, pdf = "qcReport.pdf",
    maxSamplesPerPage = 24, controls = c("BISULFITE CONVERSION I",
    "BISULFITE CONVERSION II", "EXTENSION", "HYBRIDIZATION",
    "NON-POLYMORPHIC", "SPECIFICITY I", "SPECIFICITY II", "TARGET REMOVAL"))
```

Arguments

`rgSet`	An object of class `RGChannelSet`.
`sampNames`	Sample names to be used for labels.
`sampGroups`	Sample groups to be used for labels.

`pdf` Path and name of the PDF output file.

`maxSamplesPerPage` Maximum number of samples to plot per page in those sections that plot each sample separately.

`controls` The control probe types to include in the report.

Details

This function produces a QC report as a PDF file. It is a useful first step after reading in a new dataset to get an overview of quality and to flag potentially problematic samples.

Value

No return value. A PDF is produced as a side-effect.

Author(s)

Martin Aryee <aryee@jhu.edu>.

See Also

mdsPlot, controlStripPlot, densityPlot, densityBeanPlot

Examples

```
if (require(minfiData)) {

names <- pData(RGsetEx)$Sample_Name
groups <- pData(RGsetEx)$Sample_Group

## Not run:
qcReport(RGsetEx, sampNames=names, sampGroups=groups, pdf="qcReport.pdf")

## End(Not run)

}
```

---

ratioConvert-methods *Converting methylation signals to ratios (Beta or M-values)*

Description

Converting methylation data from methylation and unmethylation channels, to ratios (Beta and M-values).

Usage

```
## S4 method for signature 'MethylSet'
ratioConvert(object, what = c("beta", "M", "both"), keepCN = TRUE, ...)
## S4 method for signature 'GenomicMethylSet'
ratioConvert(object, what = c("beta", "M", "both"), keepCN = TRUE, ...)
```

Arguments

`object`	Either a `MethylSet`, or a `GenomicRatioSet`.
`what`	Which ratios should be computed and stored?
`keepCN`	A logical, should copy number values be computed and stored in the object?
`...`	Passed to `getBeta`, `getM` methods.

Value

An object of class `RatioSet` or `GenomicRatioSet`.

Author(s)

Kasper Daniel Hansen <khansen@jhsph.edu>

See Also

`RatioSet` or codeGenomicRatioSet for the output object and `MethylSet` or codeGenomicMethylSet for the input object.

Examples

```
if (require(minfiData)) {
  ## MsetEx.sub is a small subset of MsetEx;
  ## only used for computational speed.
  RsetEx.sub <- ratioConvert(MsetEx.sub, keepCN = TRUE)
  }
```

RatioSet-class *RatioSet instances*

Description

This class holds preprocessed data for Illumina methylation microarrays.

Usage

```
## Constructor

RatioSet(Beta = NULL, M = NULL, CN = NULL,
        annotation = "", preprocessMethod = "", ...)

## Data extraction / Accessors

## S4 method for signature 'RatioSet'
getBeta(object)
## S4 method for signature 'RatioSet'
getM(object)
## S4 method for signature 'RatioSet'
getCN(object)
## S4 method for signature 'RatioSet'
preprocessMethod(object)
## S4 method for signature 'RatioSet'
annotation(object)
## S4 method for signature 'RatioSet'
pData(object)
## S4 method for signature 'RatioSet'
sampleNames(object)
## S4 method for signature 'RatioSet'
featureNames(object)
```

Arguments

`object`	A `RatioSet`.
`Beta`	A matrix of beta values (between zero and one) with each row being a methylation loci and each column a sample.
`M`	A matrix of log-ratios (between minus infinity and infinity) with each row being a methylation loci and each column a sample.
`CN`	An optional matrix of copy number estimates with each row being a methylation loci and each column a sample.
`annotation`	An annotation string, optional.
`preprocessMethod`	A `character`, optional.
`...`	For the constructor, additional arguments to be passed to `SummarizedExperiment`; of particular interest are `colData`, `rowData` and `metadata`. For `getM` these values gets passed onto `getBeta`.

Details

This class inherits from `eSet`. Essentially the class is a representation of a `Beta` matrix and/or a `M` matrix and optionally a `CN` (copy number) matrix linked to a `pData` data frame.

In addition, an annotation and a preprocessMethod slot is present. The annotation slot describes the type of array and also which annotation package to use. The preprocessMethod slot describes the kind of preprocessing that resulted in this dataset.

For a `RatioSet`, M-values are defined as `logit2` of the Beta-values if the M-values are not present in the object. Similarly, if only M-values are present in the object, Beta-values are `ilogit2` of the M-values.

Value

An object of class `RatioSet` for the constructor.

Constructor

Instances are constructed using the `RatioSet` function with the arguments outlined above.

Accessors

In the following code, `object` is a `RatioSet`.

`getBeta(object)`, `getM(object)`, `CN(object)` Get the Beta, M or CN matrix.

`getManifest(object)` get the manifest associated with the object.

`preprocessMethod(object)` Get the preprocess method `character`.

Utilities

In the following code, `object` is a `RatioSet`.

`combine`: Combines two different `RatioSet`, eventually using the `combine` method for `eSet`.

Author(s)

Kasper Daniel Hansen <khansen@jhsph.edu>

See Also

`eSet` for the basic class structure. Objects of this class are typically created from an `MethylSet` using `ratioConvert`.

Examples

```
showClass("RatioSet")
```

read.metharray *Parsing IDAT files from Illumina methylation arrays.*

Description

Parsing IDAT files from Illumina methylation arrays.

Usage

```
read.metharray(basenames, extended = FALSE, verbose = FALSE, force = FALSE)
```

Arguments

basenames	The basenames or filenames of the IDAT files. By basenames we mean the filename without the ending `_Grn.idat` or `_Red.idat` (such that each sample occur once). By filenames we mean filenames including `_Grn.idat` or `_Red.idat` (but only one of the colors)
extended	Should a `RGChannelSet` or a `RGChannelSetExtended` be returned.
verbose	Should the function be verbose?
force	Should reading different size IDAT files be forced? See Details.

Details

The type of methylation array is guess by looking at the number of probes in the IDAT files.

We have seen IDAT files from the same array, but with different number of probes in the wild. Specifically this is the case for early access EPIC arrays which have fewer probes than final release EPIC arrays. It is possible to combine IDAT files from the same inferred array, but with different number of probes, into the same `RGChannelSet` by setting `force=TRUE`. The output object will have the same number of probes as the smallest array being parsed; effectively removing probes which could have been analyzed.

Value

An object of class `RGChannelSet` or `RGChannelSetExtended`.

Author(s)

Kasper Daniel Hansen<khansen@jhsph.edu>.

See Also

`read.metharray.exp` for a convenience function for reading an experiment, `read.metharray.sheet` for reading a sample sheet and `RGChannelSet` for the output class.

Examples

```
if(require(minfiData)) {

baseDir <- system.file("extdata", package = "minfiData")
RGset1 <- read.metharray(file.path(baseDir, "5723646052", "5723646052_R02C02"))

}
```

`read.metharray.exp` *Reads an entire metharray experiment using a sample sheet*

Description

Reads an entire methylation array experiment using a sample sheet or (optionally) a target like data.frame.

Usage

```
read.metharray.exp(base = NULL, targets = NULL, extended = FALSE,
    recursive = FALSE, verbose = FALSE, force = FALSE)
```

Arguments

`base`	The base directory.
`targets`	A targets `data.frame`, see details
`extended`	Should the output of the function be a `"RGChannelSetExtended"` (default is `"RGChannelSet"`).
`recursive`	Should the search be recursive (see details)
`verbose`	Should the function be verbose?
`force`	Should reading different size IDAT files be forced? See the documentation for `read.metharray`

Details

If the `targets` argument is `NULL`, the function finds all two-color IDAT files in the directory given by `base`. If `recursive` is `TRUE`, the function searches `base` and all subdirectories. A two-color IDAT files are pair of files with names ending in `_Red.idat` or `_Grn.idat`.

If the `targets` argument is not `NULL` it is assumed it has a columned named `Basename`, and this is assumed to be pointing to the base name of a two color IDAT file, ie. a name that can be made into a real IDAT file by appending either `_Red.idat` or `_Grn.idat`.

The type of methylation array is guess by looking at the number of probes in the IDAT files.

Value

An object of class `"RGChannelSet"` or `"RGChannelSetExtended"`.

Author(s)

Kasper Daniel Hansen <khansen@jhsph.edu>.

See Also

read.metharray for the workhorse function, read.metharray.sheet for reading a sample sheet and RGChannelSet for the output class.

Examples

```
if(require(minfiData)) {

baseDir <- system.file("extdata", package = "minfiData")
RGset <- read.metharray.exp(file.path(baseDir, "5723646052"))

}
```

---

read.metharray.sheet *Reading an Illumina methylation sample sheet*

Description

Reading an Illumina methylation sample sheet, containing pheno-data information for the samples in an experiment.

Usage

```
read.metharray.sheet(base, pattern = "csv$", ignore.case = TRUE,
    recursive = TRUE, verbose = TRUE)
```

Arguments

base	The base directory from which the search is started.
pattern	What pattern is used to identify a sample sheet file, see list.files
ignore.case	Should the file search be case sensitive?
recursive	Should the file search be recursive, see list.files?
verbose	Should the function be verbose?

Details

This function search the directory base (possibly including subdirectories depending on the argument recursive for "sample sheet" files (see below). These files are identified solely on the base of their filename given by the arguments pattern and ignore.case (note the use of a dollarsign to mean end of file name).

In case multiple sheet files are found, they are all read and the return object will contain the concatenation of the files.

A sample sheet file is essentially a CSV (comma-separated) file containing one line per sample, with a number of columns describing pheno-data or other important information about the sample. The file may contain a header, in which case it is assumed that all lines up to and including a line starting with \[Data\] should be dropped. This is modelled after a sample sheet file Illumina provides. It is also very similar to the targets file made used by the popular limma package (see the extensive package vignette).

An attempt at guessing the file path to the IDAT files represented in the sheet is made. This should be doublechecked and might need to manually changed.

The type of methylation array is guess by looking at the number of probes in the IDAT files.

Value

A data.frame containing the columns of all the sample sheets. As described in details, a column named Sentrix_Position is renamed to Array and Sentrix_ID is renamed to Slide. In addition the data.frame will contain a column named Basename.

Author(s)

Kasper Daniel Hansen<khansen@jhsph.edu>.

See Also

read.metharray.exp and read.metharray for functions reading IDAT files. list.files for help on the arguments recursive and ignore.case.

Examples

```
if(require(minfiData)) {

baseDir <- system.file("extdata", package = "minfiData")
sheet <- read.metharray.sheet(baseDir)

}
```

---

readGEORawFile *Read in Unmethylated and Methylated signals from a GEO raw file.*

Description

Read in Unmethylated and Methylated signals from a GEO raw file.

Usage

```
readGEORawFile(filename, sep = ",", Uname = "Unmethylated signal",
               Mname = "Methylated signal", row.names = 1, pData = NULL,
               array = "IlluminaHumanMethylation450k",
               annotation = .default.450k.annotation, mergeManifest = FALSE,
               showProgress = TRUE, ...)
```

Arguments

filename	The name of the file to be read from.
sep	The field separator character. Values on each line of the file are separated by this character.
Uname	A string that uniquely identifies the columns containing the unmethylated signals.
Mname	A string that uniquely identifies the columns containing the methylated signals.
row.names	The column containing the feature (CpG) IDs.
pData	A DataFrame or data.frame describing the samples represented by the columns of mat. If the rownames of the pData don't match the colnames of mat these colnames will be changed. If pData is not supplied, a minimal DataFrame is created.
array	Array name.
annotation	The feature annotation to be used. This includes the location of features thus depends on genome build.
mergeManifest	Should the Manifest be merged to the final object.
showProgress	TRUE displays progress on the console. It is produced in fread's C code.
...	Additional arguments passed to data.table::fread().

Details

450K experiments uploaded to GEO typically include a raw data file as part of the supplementary materials. Unfortunately there does not appear to be a standard format. This function provides enough flexibility to read these files. Note that you will likely need to change the sep, Uname, and Mname arguments and make sure the first column includes the feature (CpG) IDs. You can use the readLines function to decipher how to set these arguments.

Note that the function uses the fread function in the **data.table** package to read the data. To install **data.table** type install.packages("data.table"). We use this package because the files too large for read.table.

Value

A GenomicMethylSet object.

Author(s)

Rafael A. Irizarry<rafa@jimmy.harvard.edu>.

See Also

getGenomicRatioSetFromGEO

Examples

```
## Not run:
library(GEOquery)
getGEOSuppFiles("GSE29290")
gunzip("GSE29290/GSE29290_Matrix_Signal.txt.gz")
# NOTE: This particular example file uses a comma as the decimal separator
#       (e.g., 0,00 instead of 0.00). We replace all such instances using the
#       command line tool 'sed' before reading in the modified file.
cmd <- paste0("sed s/,/\./g GSE29290/GSE29290_Matrix_Signal.txt > ",
              "GSE29290/GSE29290_Matrix_Signal_mod.txt")
system(cmd)
gmset <- readGEORawFile(filename = "GSE29290/GSE29290_Matrix_Signal_mod.txt",
                        Uname = "Signal_A",
                        Mname = "Signal_B",
                        sep = "\t")

## End(Not run)
```

---

readTCGA *Read in tab deliminited file in the TCGA format*

Description

Read in tab deliminited file in the TCGA format

Usage

```
readTCGA(filename, sep = "\t", keyName = "Composite Element REF", Betaname = "Beta_value",
         pData = NULL, array = "IlluminaHumanMethylation450k",
         annotation = .default.450k.annotation, mergeManifest = FALSE,
         showProgress = TRUE)
```

Arguments

filename	The name of the file to be read from.
sep	The field separator character. Values on each line of the file are separated by this character.
keyName	The column name of the field containing the feature IDs.
Betaname	The character string contained all column names of the beta value fields.
pData	A DataFrame or data.frame describing the samples represented by the columns of mat. If the rownames of the pData don't match the colnames of mat these colnames will be changed. If pData is not supplied, a minimal DataFrame is created.
array	Array name.
annotation	The feature annotation to be used. This includes the location of features thus depends on genome build.
mergeManifest	Should the Manifest be merged to the final object.
showProgress	TRUE displays progress on the console. It is produced in fread's C code.

Details

This function is a wrapper for `makeGenomicRatioSetFromMatrix`. It assumes a very specific format, used by TCGA, and then uses the `fread` function in the **data.table** package to read the data. To install **data.table** type `install.packages("data.table")`. We use this package because the files too large for `read.table`.

Currently, an example of a file that this function reads is here: `http://gdac.broadinstitute.org/runs/stddata__2014_10_17/data/UCEC/20141017/gdac.broadinstitute.org_UCEC.Merge_methylation__humanmethylation450__jhu_usc_edu__Level_3__within_bioassay_data_set_function__data.Level_3.2014101700.0.0.tar.gz`. Note it is a 8.1 GB archive.

Value

A GenomicRatioSet object.

Author(s)

Rafael A. Irizarry`<rafa@jimmy.harvard.edu>`.

See Also

`makeGenomicRatioSetFromMatrix`

Examples

```
## Not run:
    filename <- "example.txt" ##file must be in the specicif TCGA format
    readTCGA(filename)

## End(Not run)
```

RGChannelSet-class   *Class* "RGChannelSet"

Description

These classes represents raw (unprocessed) data from a two color micro array; specifically an Illumina methylation array.

Usage

```
## Constructors

RGChannelSet(Green = new("matrix"), Red = new("matrix"),
             annotation = "", ...)

RGChannelSetExtended(Green = new("matrix"), Red = new("matrix"),
                     GreenSD = new("matrix"), RedSD = new("matrix"),
```

```
                    NBeads = new("matrix"), annotation = "", ...)

## Accessors

## S4 method for signature 'RGChannelSet'
annotation(object)
## S4 method for signature 'RGChannelSet'
pData(object)
## S4 method for signature 'RGChannelSet'
sampleNames(object)
## S4 method for signature 'RGChannelSet'
featureNames(object)
## S4 method for signature 'RGChannelSet'
getBeta(object, ...)
getGreen(object)
getRed(object)
getNBeads(object)
## S4 method for signature 'RGChannelSet'
getManifest(object)

## Convenience functions
getOOB(object)
getSnpBeta(object)
```

Arguments

object	An RGChannelSet (or RGChannelSetExtended).
Green	A matrix of Green channel values (between zero and infinity) with each row being a methylation loci and each column a sample.
Red	See the Green argument, but for the Green channel.
GreenSD	See the Green argument, but for standard deviations of the Green channel summaries.
RedSD	See the Green, but for standard deviations of the Red channel summaries.
NBeads	See the Green argument, but contains the number of beads used to summarize the Green and Red channels.
annotation	An annotation string, optional.
...	For the constructor(s), additional arguments to be passed to SummarizedExperiment; of particular interest are colData, rowData and metadata. For getBeta these values gets passed onto getBeta.

Value

An object of class RGChannelSet or RGChannelSetExtended for the constructors.

Constructors

Instances are constructed using the RGChannelSet or RGChannelSetExtended functions with the arguments outlined above.

as(object, "RGChannelSet") coerces a RGChannelSetExtended object into a RGChannelSet.

Accessors

getGreen: Gets the Green channel as a matrix.

getRed: Gets the Red channel as a matrix.

getNBeads: Gets the number of beads as a matrix, this requires an RGChannelSetExtended.

getManifest: Gets the manifest object itself associated with the array type

Convenience functions

getOOB: Retrives the so-called "out-of-band" (OOB) probes. These are the measurements of Type I probes in the "wrong" color channel. Return value is a list with two matrices, named Red and Grn.

getSnpBeta: Retrives the measurements of the 65 SNP probes located on the array. These SNP probes are intended to be used for sample tracking and sample mixups. The return value is a matrix of beta values. Each SNP probe ought to have values clustered around 3 distinct values corresponding to homo-, and hetero-zygotes.

combine: Combines two different RGChannelSet, eventually using the combine method for eSet.

Tips

The class inherits a number of useful methods from SummarizedExperiment. In earliers versions of minfi, this class inherited from eSet, and we have kept of number of methods related to this, for example pData.

The best way to access phenotype data and sample names are colData and colnames.

Amongst the useful methods are

dim, nrow, ncol The dimension (number of probes by number of samples) of the experiment.

colData, colnames, pData, sampleNames Phenotype information and sample names.

rownames, featureNames This is the addresses (probe identifiers) of the array.

Author(s)

Kasper Daniel Hansen <khansen@jhsph.edu>

See Also

See SummarizedExperiment for the basic class that is used as a building block for "RGChannelSet(Extended)". See IlluminaMethylationManifest for a class representing the design of the array.

Examples

```
showClass("RGChannelSet")
```

subsetByLoci *Subset an RGChannelset by CpG loci.*

Description

Subset an RGChannelSet by CpG loci.

Usage

```
subsetByLoci(rgSet, includeLoci = NULL, excludeLoci = NULL,
             keepControls = TRUE, keepSnps = TRUE)
```

Arguments

`rgSet`	An object of class `RGChannelSet` (or `RGChannelSetExtended`).
`includeLoci`	A character vector of CpG identifiers which should be kept.
`excludeLoci`	A character vector of CpG identifiers which should be excluded.
`keepControls`	Should control probes be kept?
`keepSnps`	Should SNP probes be kept?

Details

This task is non-trivial because an `RGChannelSet` is indexed by probe position on the array, not by loci name.

Value

An object of class `RGChannelSet`, which some probes removed.

Examples

```
if(require(minfiData)) {
  loci <- c("cg00050873", "cg00212031", "cg00213748", "cg00214611")
  subsetByLoci(RGsetEx.sub, includeLoci = loci)
  subsetByLoci(RGsetEx.sub, excludeLoci = loci)
}
```

Index