Package ‘minfi’

April 5, 2014

Version 1.8.9

Title Analyze Illumina's 450k methylation arrays

Description Tools for analyzing and visualizing Illumina's 450k array data

Depends methods, BiocGenerics (>= 0.3.2), Biobase (>= 2.17.8),lattice, reshape, GenomicRanges, Biostrings, utils, bumphunter (>= 1.1.9)

Suggests IlluminaHumanMethylation450kmanifest (>= 0.2.0),IlluminaHumanMethylation450kanno.ilmn12.hg19, minfiData (>= 0.4.1), quadprog, FlowSorted.Blood.450k (>= 1.0.1), RUnit,digest

Imports IRanges, beanplot, RColorBrewer, nor1mix, siggenes, limma,preprocessCore, illuminaio, matrixStats, mclust, genefilter,nlme

Collate generics.R mset.R rset.R rgset.R gmset.R grset.R utils.R manifest.R anno.R minfiQC.R getSex.R dmr.R blocks.R plot.R plotBetasByType.R preprocess.R preprocessSwan.R preprocessQuantile.R qc.R read.450k.R read.manifest.R bumphunter.R estimateCellCounts.R

License Artistic-2.0

LazyData yes

biocViews DNAMethylation, Microarray, TwoChannel, DataImport,Preprocessing, QualityControl

Author Kasper Daniel Hansen [cre, aut],Martin Ayree [aut],Rafael A. Irizarry [aut],Andrew E. Jaffe [ctb],Jovana Maksimovic [ctb],E. Andres Houseman [ctb]

Maintainer Kasper Daniel Hansen <khansen@jhsph.edu>

R topics documented:

minfi-package 3
blockFinder 3
bumphunter-methods 4
controlStripPlot 7
cpgCollapse 8
densityBeanPlot 9
densityPlot 10
detectionP 11
dmpFinder 12
estimateCellCounts 14
fixMethOutliers 16
GenomicMethylSet-class 17
GenomicRatioSet-class 19
getAnnotation 21
getMethSignal 22
getQC 23
getSex 24
IlluminaMethylationAnnotation-class 25
IlluminaMethylationManifest-class 26
logit2 28
mapToGenome-methods 29
mdsPlot 30
MethylSet-class 31
minfiQC 33
plotBetasByType 34
plotCpg 36
preprocessIllumina 37
preprocessQuantile 38
preprocessRaw 40
preprocessSWAN 41
qcReport 42
ratioConvert-methods 44
RatioSet-class 44
read.450k 46
read.450k.exp 47
read.450k.sheet 48
RGChannelSet-class 49

Index **52**

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,
            smooth = TRUE, smoothFunction = locfitByCluster,
            B = 1000, 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?
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: 1000 is recommended but can be quite slow if this process is not parallelized.
verbose	Should the function be verbose?
bpSpan	Smoothing span
...	Currently unused

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, cutoffQ = 0.99,
          maxGap=500, smooth=FALSE, smoothFunction=loessByCluster,
          useWeights=FALSE, B=1000, verbose=TRUE,
          type = c("M", "Beta"), ...)
```

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.
`cutoffQ`	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.
`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: 1000 is recommended but can be quite slow if this process is not parallelized.
`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

This function performs the bumphunting approach described by Jaffe et al. International Journal of Epidemiology (2012). The main output is a table of candidate regions with permutation-based family-wide error rates (FWER) and p-values assigned.

The general idea is that for each genomic location we have a value for several individuals. We also have covariates for each individual and perform regression. This gives us one estimate of the coefficient of interest (a common example is case versus control). These estimates are then (optionally) smoothed. The smoothing occurs in clusters of locations that are ‘close enough’. This gives us an estimate of a genomic profile that is 0 when uninteresting. We then take values above (in absolute value) `cutoff` as candidate regions. Permutations are then performed to create null distributions for the candidate regions.

Uncertainty is assessed via permutations. Each of the B permutations will produce an estimated 'null profile' from which we can define 'null candidate regions'. For each observed candidate region we determine how many null regions are 'more extreme' (longer and higher average value). The 'p.value' is the percent of candidate regions obtained from the permutations that as extreme as the observed region. These p-values should be interpreted with care as the theoretical proporties are not well understood. The 'fwer' is the proportion of permutations that had at least on regio as extreme as the observed region. We compute p.values and FWER for the area of the regions (as opposed to length and value as a pair) as well.

Parallelization is implemented through the foreach package.

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

Jaffe AE, Murakami P, Lee H, Leek JT, Fallin MD, Feinberg AP, Irizarry RA (2012) Bump hunting to identify differentially methylated regions in epigenetic epidemiology studies. International Journal of Epidemiology 41(1):200-9.

Examples

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

`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)

}
```

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`.
`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`

`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

Kampstra, P. 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)
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",
                   cellTypes = c("CD8T","CD4T", "NK","Bcell","Mono","Gran"),
                   returnAll = FALSE, meanPlot = FALSE, verbose = TRUE, ...)
```

Arguments

rgSet	The input RGChannelSet for the procedure.
meanPlot	Whether to plots the average DNA methylation across the cell-type discrimating probes within the mixed and sorted samples.
compositeCellType	Which composite cell type is being deconvoluted, see details.
cellTypes	Which celltypes, from the reference object, should be we use for the deconvolution?
returnAll	Should the composition table and the normalized user supplied data be return?
verbose	Should the function be verbose?
...	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 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.

Note

At the time of writing, only data from deconvolution of whole blood cells are available.

Author(s)

Andrew E. Jaffe, E. Andres Houseman

References

Houseman, Eugene A., et al. "DNA methylation arrays as surrogate measures of cell mixture distribution." BMC bioinformatics 13.1 (2012): 86.

Andrew E. Jaffe and Rafael A. Irizarry. Accounting for cellular heterogeneity is critical in epigenome-wide association studies. Under Review.

See Also

`preprocessQuantile`

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)
}
```

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"), pData = DataFrame(),
                 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.

- `pData` A `DataFrame` or `data.frame` 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 `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`.

Constructor

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

Accessors

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

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.

Author(s)

Kasper Daniel Hansen <khansen@jhsph.edu>

See Also

SummarizedExperiment 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, pData = DataFrame(),
                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.
pData	A DataFrame or data.frame object.
annotation	An annotation character string.
preprocessMethod	A preprocess method character string.
...	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.

Constructor

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

Accessors

A number of useful accessors are inherited from SummarizedExperiment.

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.

Author(s)

Kasper Daniel Hansen <khansen@jhsph.edu>

See Also

SummarizedExperiment 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)

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

getProbeType(object)

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.
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.

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.

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

Value

For getAnnotation, a DataFrame with the requested information.

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 elementMetadata of the granges of the object.

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")
}
```

---

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)) {
  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 a column named predictedSex added to the pheno data.

Author(s)

Rafael A. Irizarry, Kasper D. Hansen

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(listOfObjects, annotation = "",
                              defaults = "")

## 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")`.
`listOfObjects`	A list of objects to be put into the data slot of the annotation object.

Details

The `data` slot contains various objects that are of type `data.frame`. Details are still subject to change.

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 = new("DataFrame"),
                            TypeII = new("DataFrame"),
                            TypeControl = new("DataFrame"),
                            TypeSnpI = new("DataFrame"),
                            TypeSnpII = new("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"))
getControlTypes(object)
```

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.

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 `data.frame` 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.

`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))
```

`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.

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.

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

Borg, I., Groenen, P. (2005). Modern Multidimensional Scaling: theory and applications (2nd ed.). New York: Springer-Verlag. 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"),  ...)

## 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)

## 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.
- `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 objects passes to the `eSet` constructor, particular a `phenoData` slot. 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

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

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

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

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.

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).

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")
```

`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")

}
```

---

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]

}
```

---

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

Author(s)

Rafael A. Irizarry

References

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

See Also

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

Examples

```
if(require(minfiData)) {
  GMset <- preprocessQuantile(RGsetEx)
}
```

---

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

```
## 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 compy number values be computed and stored in the object?
`...`	Passed to `getBeta`, `getM` methods.

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.

`RatioSet-class` *RatioSet instances*

Description

This class holds preprocessed data for Illumina methylation microarrays.

Usage

```
## Constructor

RatioSet(Beta = NULL, M = NULL, CN = NULL,  ...)

## 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)
```

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.
`...`	For the constructor: additional objects passes to the `eSet` constructor, particular a `phenoData` slot. 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.

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`.

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.450k` *Parsing IDAT files from Illumina methylation arrays.*

Description

Parsing IDAT files from Illumina methylation arrays.

Usage

```
read.450k(basenames, extended = FALSE, verbose = 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?

Value

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

Author(s)

Kasper Daniel Hansen<khansen@jhsph.edu>.

See Also

read.450k.exp for a convenience function for reading an experiment, read.450k.sheet for reading a sample sheet and RGChannelSet for the output class.

Examples

```
if(require(minfiData)) {

baseDir <- system.file("extdata", package = "minfiData")
RGSet <- read.450k(file.path(baseDir, "5723646052",
    c("5723646052_R02C02", "5723646052_R04C01")))

}
```

---

read.450k.exp — *Reads an entire 450k experiment using a sample sheet*

Description

Reads an entire 450k experiment using a sample sheet or (optionally) a target like data.frame.

Usage

```
read.450k.exp(base, targets = NULL, extended = FALSE,
    recursive = FALSE, verbose = 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?

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.

Value

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

Author(s)

Kasper Daniel Hansen <khansen@jhsph.edu>.

See Also

read.450k for the workhorse function, read.450k.sheet for reading a sample sheet and RGChannelSet for the output class.

Examples

```
if(require(minfiData)) {

baseDir <- system.file("extdata", package = "minfiData")
RGset <- read.450k.exp(file.path(baseDir, "5723646052"))

}
```

---

read.450k.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.450k.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.

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.450k.exp` and `read.450k` 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.450k.sheet(baseDir)

}
```

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

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

## Accessors

## S4 method for signature RGChannelSet
getBeta(object, ...)
getGreen(object)
getRed(object)
## S4 method for signature RGChannelSet
getManifest(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.
`...`	Additional objects passes to the `eSet` constructor, particular a `phenoData` slot.

Details

FIXME

Constructors

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

Accessors

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

`getGreen`**:** Gets the Green channel as a matrix.

`getRed`**:** Gets the Red channel as a matrix.

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

Tips

The class inherits a number of useful methods from `eSet`. Amongst these are

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

`pData`, `sampleNames` Phenotype information and sample names.

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

Author(s)

Kasper Daniel Hansen <khansen@jhsph.edu>

See Also

See `eSet` 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")
```

Index

∗Topic **classes**
IlluminaMethylationAnnotation-class, 25
IlluminaMethylationManifest-class, 26
∗Topic **methods**
mapToGenome-methods, 29
ratioConvert-methods, 44
∗Topic **package**
minfi-package, 3

addQC *(*getQC*)*, 23
addSex *(*getSex*)*, 24
addSnpInfo *(*getAnnotation*)*, 21
annotation,GenomicMethylSet-method *(*GenomicMethylSet-class*)*, 17
annotation,GenomicRatioSet-method *(*GenomicRatioSet-class*)*, 19

bgcorrect.illumina *(*preprocessIllumina*)*, 37
blockFinder, 3, *9*
bumphunter, *4*
bumphunter,GenomicRatioSet-method *(*bumphunter-methods*)*, 4
bumphunter-methods, 4

clusterMaker, *3*, *5*
controlStripPlot, 7, *10*, *11*, *31*, *43*
cpgCollapse, *4*, 8

densityBeanPlot, *7*, 9, *11*, *31*, *35*, *43*
densityPlot, *7*, 10, *10*, *31*, *35*, *43*
detectionP, 11
dmpFinder, 12
dropMethylationLoci *(*MethylSet-class*)*, 31

eSet, *33*, *46*, *51*
estimateCellCounts, 14

featureNames,GenomicMethylSet-method *(*GenomicMethylSet-class*)*, 17
featureNames,GenomicRatioSet-method *(*GenomicRatioSet-class*)*, 19
fixMethOutliers, 16, *34*, *40*

GenomicMethylSet, *29*, *44*
GenomicMethylSet *(*GenomicMethylSet-class*)*, 17
GenomicMethylSet-class, 17
GenomicRatioSet, *44*
GenomicRatioSet *(*GenomicRatioSet-class*)*, 19
GenomicRatioSet-class, 19
getAnnotation, 21
getBeta *(*MethylSet-class*)*, 31
getBeta,GenomicMethylSet-method *(*GenomicMethylSet-class*)*, 17
getBeta,GenomicRatioSet-method *(*GenomicRatioSet-class*)*, 19
getBeta,MethylSet-method *(*MethylSet-class*)*, 31
getBeta,RatioSet-method *(*RatioSet-class*)*, 44
getBeta,RGChannelSet-method *(*RGChannelSet-class*)*, 49
getCN *(*GenomicRatioSet-class*)*, 19
getCN,GenomicMethylSet-method *(*GenomicMethylSet-class*)*, 17
getCN,GenomicRatioSet-method *(*GenomicRatioSet-class*)*, 19
getCN,MethylSet-method *(*MethylSet-class*)*, 31
getCN,RatioSet-method *(*RatioSet-class*)*, 44
getControlAddress *(*IlluminaMethylationManifest-class*)*, 26
getControlTypes *(*IlluminaMethylationManifest-class*)*,

26
getGreen (RGChannelSet-class), 49
getIslandStatus (getAnnotation), 21
getLocations, *29*
getLocations (getAnnotation), 21
getM (MethylSet-class), 31
getM,GenomicMethylSet-method
(GenomicMethylSet-class), 17
getM,GenomicRatioSet-method
(GenomicRatioSet-class), 19
getM,MethylSet-method
(MethylSet-class), 31
getM,RatioSet-method (RatioSet-class),
44
getManifest
(IlluminaMethylationManifest-class),
26
getManifest,character-method
(IlluminaMethylationManifest-class),
26
getManifest,IlluminaMethylationAnnotation-method
(IlluminaMethylationAnnotation-class),
25
getManifest,IlluminaMethylationManifest-method
(IlluminaMethylationManifest-class),
26
getManifest,MethylSet-method
(MethylSet-class), 31
getManifest,RGChannelSet-method
(RGChannelSet-class), 49
getManifestInfo
(IlluminaMethylationManifest-class),
26
getMeth (MethylSet-class), 31
getMeth,GenomicMethylSet-method
(GenomicMethylSet-class), 17
getMeth,MethylSet-method
(MethylSet-class), 31
getMethSignal, 22
getProbeInfo
(IlluminaMethylationManifest-class),
26
getProbeType (getAnnotation), 21
getQC, 23, *34*
getRed (RGChannelSet-class), 49
getSex, 24, *34*, *40*
getSnpInfo (getAnnotation), 21
getUnmeth (MethylSet-class), 31
getUnmeth,GenomicMethylSet-method
(GenomicMethylSet-class), 17
getUnmeth,MethylSet-method
(MethylSet-class), 31

IlluminaMethylationAnnotation, *22*, *27*
IlluminaMethylationAnnotation
(IlluminaMethylationAnnotation-class),
25
IlluminaMethylationAnnotation-class,
25
IlluminaMethylationManifest, *26*, *38*, *40*,
42, *51*
IlluminaMethylationManifest
(IlluminaMethylationManifest-class),
26
IlluminaMethylationManifest-class, 26
ilogit2 (logit2), 28
initialize,MethylSet-method
(MethylSet-class), 31
initialize,RatioSet-method
(RatioSet-class), 44
initialize,RGChannelSet-method
(RGChannelSet-class), 49
initialize,RGChannelSetExtended-method
(RGChannelSet-class), 49

legend, *30*, *31*, *35*
list.files, *49*
logit2, 28

mapToGenome, *19*, *22*
mapToGenome (mapToGenome-methods), 29
mapToGenome,GenomicMethylSet-method
(GenomicMethylSet-class), 17
mapToGenome,GenomicRatioSet-method
(GenomicRatioSet-class), 19
mapToGenome,MethylSet-method
(mapToGenome-methods), 29
mapToGenome,RatioSet-method
(mapToGenome-methods), 29
mapToGenome,RGChannelSet-method
(mapToGenome-methods), 29
mapToGenome-methods, 29
mdsPlot, *7*, *10*, *11*, 30, *43*
MethylSet, *18*, *19*, *29*, *38*, *40*, *42*, *44*, *46*
MethylSet (MethylSet-class), 31
MethylSet-class, 31
minfi (minfi-package), 3

minfi-package, 3
minfiQC, *16*, *24*, 33, *40*

normalize.illumina.control (preprocessIllumina), 37

par, *30, 31*, *35*
pData,GenomicMethylSet-method (GenomicMethylSet-class), 17
pData,GenomicRatioSet-method (GenomicRatioSet-class), 19
plotBetasByType, 34
plotCpg, 36
plotQC (getQC), 23
plotSex (getSex), 24
preprocessIllumina, 37
preprocessMethod (MethylSet-class), 31
preprocessMethod,GenomicMethylSet-method (GenomicMethylSet-class), 17
preprocessMethod,GenomicRatioSet-method (GenomicRatioSet-class), 19
preprocessMethod,MethylSet-method (MethylSet-class), 31
preprocessMethod,RatioSet-method (RatioSet-class), 44
preprocessQuantile, *15*, 38
preprocessRaw, *33*, *38*, 40
preprocessSWAN, *35*, 41

qcReport, *7*, *10, 11*, *31*, 42

ratioConvert, *46*
ratioConvert (ratioConvert-methods), 44
ratioConvert,GenomicMethylSet-method (ratioConvert-methods), 44
ratioConvert,MethylSet-method (ratioConvert-methods), 44
ratioConvert-methods, 44
RatioSet, *44*
RatioSet (RatioSet-class), 44
RatioSet-class, 44
read.450k, 46, *48, 49*
read.450k.exp, 47, *47*, *49*
read.450k.sheet, *47*, 48, *48*
RGChannelSet, *33*, *38*, *40*, *42*, *47, 48*
RGChannelSet (RGChannelSet-class), 49
RGChannelSet-class, 49
RGChannelSetExtended (RGChannelSet-class), 49
RGChannelSetExtended-class (RGChannelSet-class), 49

sampleNames,GenomicMethylSet-method (GenomicMethylSet-class), 17
sampleNames,GenomicRatioSet-method (GenomicRatioSet-class), 19
show,GenomicMethylSet-method (GenomicMethylSet-class), 17
show,GenomicRatioSet-method (GenomicRatioSet-class), 19
show,IlluminaMethylationAnnotation-method (IlluminaMethylationAnnotation-class), 25
show,IlluminaMethylationManifest-method (IlluminaMethylationManifest-class), 26
show,MethylSet-method (MethylSet-class), 31
show,RatioSet-method (RatioSet-class), 44
show,RGChannelSet-method (RGChannelSet-class), 49
squeezeVar, *13*
SummarizedExperiment, *19*, *21*

xy.coords, *35*