Package 'minfi'

March 26, 2013

Version 1.4.0

Title Analyze Illumina's 450k methylation arrays

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

Author Kasper Daniel Hansen, Martin Aryee

Maintainer Kasper Daniel Hansen <khansen@jhsph.edu>

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

Suggests IlluminaHumanMethylation450kmanifest (>= 0.2.0), minfiData (>= 0.2.0), RUnit

Imports BiocGenerics, beanplot, RColorBrewer, nor1mix, siggenes,limma, preprocessCore, crlmm, matrixStats, mclust

Collate generics.R rgset.R mset.R gmset.R utils.R locations.R anno.R dmr.R plot.R plotBetasByType.R preprocess.R preprocessSwan.R qc.R read.450k.R read.manifest.R

License Artistic-2.0

LazyData yes

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

R topics documented:

---

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.

---

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)

}
```

---

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)

}
```

GenomicMethylSet-class *GenomicMethylSet instances*

Description

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

Usage

```
## Constructor

GenomicMethylSet(gr, Meth, Unmeth, pData, 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'
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)
```

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.

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.

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.

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")
```

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 = "")

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

## S4 method for signature 'IlluminaMethylationAnnotation'
getLocations(object, genomeBuild = "hg19", mergeManifest = FALSE)
```

Arguments

object	An object of class IlluminaMethylationAnnotation.
annotation	An annotation character.
listOfObjects	A list of objects to be put into the data slot of the annotation object.
genomeBuild	Which genome build to use.
mergeManifest	Should the information in the associated manifest package be merged in (added as elementMetadata in the output GRanges).

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.

getLocation FIXME

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, genomeBuild = c("hg19", "hg18"), drop = TRUE, mergeManifest = FALSE)
## S4 method for signature 'RGChannelSet'
mapToGenome(object, ...)
```

Arguments

object	Either a MethylSet or RGChannelSet.
genomeBuild	Which version of the genome to use?
drop	Should unmapped loci be dropped?
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.

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'
getManifest(object)
## S4 method for signature 'MethylSet'
preprocessMethod(object)

## Utilities
## S4 method for signature 'MethylSet'
getLocations(object, genomeBuild = "hg19", drop = TRUE, mergeManifest = FALSE)
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.
genomeBuild	Which genome build to use.
drop	Should unmapped loci be dropped?

mergeManifest	Should the information in the associated manifest package be merged in (added as elementMetadata to the output GRanges).
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

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

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.

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.

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.

getLocations(object) This function obtains the genomic locations based on an annotation package. There is the option of merging in data from the manifest package in the resulting GRanges. Finally, the result is ordered according to the order of loci in the MethylSet, unless drop=TRUE where the number of locations is less than or equal to the number of loci. Unmapped locations have a seqname of unmapped.

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")
```

---

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

```
if (require(minfiData)) {

Mset.swan <- preprocessSWAN(RGsetEx, MsetEx)

par(mfrow=c(1,2))
plotBetasByType(MsetEx[,1], main="Raw")
plotBetasByType(Mset.swan[,1], main="SWAN")
}
```

---

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]

}
```

---

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)
```

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.

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

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)) {
  dat <- preprocessRaw(RGsetEx)
  slot(name="preprocessMethod", dat)[1]
  datSwan <- preprocessSWAN(RGsetEx, mSet = dat)
  datIlmn <- preprocessIllumina(RGsetEx)
  slot(name="preprocessMethod", datIlmn)[1]
  datIlmnSwan <- preprocessSWAN(RGsetEx, mSet = datIlmn)
}
```

---

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)

}
```

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