Package 'missMethyl'

May 2, 2024

Type Package

Title Analysing Illumina HumanMethylation BeadChip Data

Version 1.38.0

Date 2020-09-16

Author Belinda Phipson and Jovana Maksimovic

Maintainer Belinda Phipson <phipson.b@wehi.edu.au>, Jovana Maksimovic <jovana.maksimovic@petermac.org>, Andrew Lonsdale <andrew.lonsdale@petermac.org>

Depends R (>= 3.6.0), IlluminaHumanMethylation450kanno.ilmn12.hg19, IlluminaHumanMethylationEPICanno.ilm10b4.hg19

Imports AnnotationDbi, BiasedUrn, Biobase, BiocGenerics, GenomicRanges, GO.db, IlluminaHumanMethylation450kmanifest, IlluminaHumanMethylationEPICmanifest, IRanges, limma, methods, methylumi, minfi, org.Hs.eg.db, ruv, S4Vectors, statmod, stringr, SummarizedExperiment

VignetteBuilder knitr

Suggests BiocStyle, edgeR, knitr, minfiData, rmarkdown, tweeDEseqCountData, DMRcate, ExperimentHub

Description Normalisation, testing for differential variability and differential methylation and gene set testing for data from Illumina's Infinium HumanMethylation arrays. The normalisation procedure is subset-quantile within-array normalisation (SWAN), which allows Infinium I and II type probes on a single array to be normalised together. The test for differential variability is based on an empirical Bayes version of Levene's test. Differential methylation testing is performed using RUV, which can adjust for systematic errors of unknown origin in high-dimensional data by using negative control probes. Gene ontology analysis is performed by taking into account the number of probes per gene on the array, as well as taking into account multi-gene associated probes.

License GPL-2

biocViews Normalization, DNAMethylation, MethylationArray, GenomicVariation, GeneticVariability, DifferentialMethylation, GeneSetEnrichment

RoxygenNote 7.1.1

Encoding UTF-8

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

git_branch RELEASE_3_19

git_last_commit ae6957a

git_last_commit_date 2024-04-30

Repository Bioconductor 3.19

Date/Publication 2024-05-01

Contents

`missMethyl-package` *Introduction to the missMethyl package*

Description

missMethyl is a library for the analysis of Illumina's 450K human methylation BeadChip. Specifically, functions for SWAN normalisation and differential variability analysis are provided. SWAN normalisation uses probe specific information, and the differential variability procedure uses linear models which can handle any designed experiment.

Details

Package:	missMethyl
Type:	Package
Version:	0.99.0
Date:	2014-06-30
License:	GPL-2

Normalisation of the 450K arrays can be performed using the function `SWAN`.

Differential variability analysis can be performed by calling `varFit` followed by `topVar` for a list of the top ranked differentially variable CpGs between conditions.

More detailed help documentation is provided in each function's help page.

Author(s)

Belinda Phipson and Jovana Maksimovic

Maintainer: Belinda Phipson <belinda.phipson@mcri.edu.au>, Jovana Maksimovic <jovana.maksimovic@petermac.org>

References

Maksimovic, J., Gordon, L., Oshlack, A. (2012). SWAN: Subset-quantile within array normalization for illumina infinium HumanMethylation450 BeadChips. *Genome Biology*, 13:R44.

Phipson, B., and Oshlack, A. (2014). DiffVar: A new method for detecting differential variability with application to methylation in cancer and aging. *Genome Biology*, **15**:465.

`contrasts.varFit` *Compute contrasts for a varFit object.*

Description

Compute estimated coefficients, standard errors and LogVarRatios for a given set of contrasts.

Usage

```
contrasts.varFit(fit, contrasts = NULL)
```

Arguments

`fit`	List containing a linear model fit produced by `varFit`. The fit object should be of class `MArrayLM`.
`contrasts`	Numeric matrix with rows corresponding to coefficients in `fit` and columns containing contrasts.

Details

This function calls the `contrasts.fit` function in `limma` to compute coefficients and standard errors for the specified contrasts corresponding to a linear model fit obtained from the `varFit` function. LogVarRatios are also computed in terms of the contrasts. A contrasts matrix can be computed using the `makeContrasts` function.

Value

A list object of the same class as `fit`.

Author(s)

Belinda Phipson

See Also

`varFit`, `contrasts.fit`, `makeContrasts`

Examples

```
# Randomly generate data for a 3 group problem with 100 CpG sites and 4
# arrays in each group.

library(limma)

y<-matrix(rnorm(1200),ncol=12)

group<-factor(rep(c(1,2,3),each=4))
design<-model.matrix(~0+group)
colnames(design)<-c("grp1","grp2","grp3")
design

# Fit linear model for differential variability
# Please always specify the coef parameter in the call to varFit
vfit<-varFit(y,design,coef=c(1,2,3))

# Specify contrasts
contr<-makeContrasts(grp2-grp1,grp3-grp1,grp3-grp2,levels=colnames(design))

# Compute contrasts from fit object
vfit.contr<-contrasts.varFit(vfit,contrasts=contr)

summary(decideTests(vfit.contr))

# Look at top table of results for first contrast
topVar(vfit.contr,coef=1)
```

densityByProbeType *Plot the beta value distributions of the Infinium I and II probe types relative to the overall beta value distribution.*

Description

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

Usage

```
densityByProbeType(
  data,
  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.
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 SWAN affects the data.

Value

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

Author(s)

Jovana Maksimovic

See Also

densityPlot, densityBeanPlot, par, legend

Examples

```
if (require(minfi) & require(minfiData)) {
  dat <- preprocessRaw(RGsetEx)
  datSwan <- SWAN(dat)
  par(mfrow=c(1,2))
  densityByProbeType(dat[,1], main="Raw")
  densityByProbeType(datSwan[,1], main="SWAN")
}
```

---

getAdj *Extract values adjusted for unwanted variation by RUVm*

Description

Extract values adjusted for unwanted variation by RUVm.

Usage

```
getAdj(Y, fit)
```

Arguments

Y	A matrix of M-values.
fit	The list list object produced by RUVfit.

Details

This function extracts values adjusted for unwanted variation by RUVm. These values are ONLY intendeded to be used for visualisation purposes. It is NOT recommended that they are used for any further analysis.

Value

An matrix of M-values.

Author(s)

Jovana Maksimovic

See Also

MArrayLM

Examples

```
if(require(minfi) & require(minfiData) & require(limma)) {

# Get methylation data for a 2 group comparison
meth <- getMeth(MsetEx)
unmeth <- getUnmeth(MsetEx)
Mval <- log2((meth + 100)/(unmeth + 100))

group <- factor(pData(MsetEx)$Sample_Group, labels=c(0,1))
design <- model.matrix(~group)

# Perform initial analysis to empirically identify negative control features
# when not known a priori
lFit <- lmFit(Mval,design)
lFit2 <- eBayes(lFit)
lTop <- topTable(lFit2,coef=2,num=Inf)

# The negative control features should *not* be associated with factor of
# interest but *should* be affected by unwanted variation
ctl <- rownames(Mval) %in% rownames(lTop[lTop$adj.P.Val > 0.5,])

# Perform RUV adjustment and fit
fit <- RUVfit(Y=Mval, X=group, ctl=ctl)
fit2 <- RUVadj(Y=Mval, fit=fit)

# get adjusted values
Madj <- getAdj(Y=Mval,fit=fit)
}
```

`getINCs` *Extract intensity data for Illumina negative controls found on 450k or EPIC arrays.*

Description

Extracts the intensity data for the Illumina negative controls found on 450k or EPIC arrays and returns a matrix of M-values (log2 ratio of the green to red intensities).

Usage

```
getINCs(rgSet)
```

Arguments

`rgSet`	An object of class `RGChannelSet`.

Details

The getINCs function extracts the intensity data for the INCs from an RGChannelSet object. The function retrieves both the green and red channel intensity values and returns the data as the log2 ratio of the green to red intensities. Essentially, the INCs are being treated like Type II probes for which the M-values are also given as the log2 ratio of the green to red intensities.

Value

An matrix of M-values.

Author(s)

Jovana Maksimovic

See Also

RGChannelSet

Examples

```
if (require(minfi) & require(minfiData)) {

  INCs <- getINCs(RGsetEx)
  head(INCs)
  dim(INCs)
}
```

`getLeveneResiduals` *Obtain Levene residuals*

Description

Obtain absolute or squared Levene residuals for each CpG given a series of methylation arrays

Usage

```
getLeveneResiduals(data, design = NULL, type = NULL)
```

Arguments

`data`	Object of class `matrix` of M values, with rows corresponding to features of interest such as CpG sites and columns corresponding to samples or arrays.
`design`	The design matrix of the experiment, with rows corresponding to arrays/samples and columns to coefficients to be estimated. Defaults to the unit vector.
`type`	Character string, `"AD"` for absolute residuals or `"SQ"` for squared residuals. Default is `"AD"`.

Details

This function will return absolute or squared Levene residuals given a matrix of M values and a design matrix. This can be used for graphing purposes or for downstream analysis such a gene set testing based on differential variability rather than differential methylation. If no design matrix is given, the residuals are determined by treating all samples as coming from one group.

Value

Returns a list with three components. `data` contains a matrix of absolute or squared residuals, `AvgVar` is a vector of sample variances and `LogVarRatio` corresponds to the columns of the design matrix and is usually the ratios of the log of the group variances.

Author(s)

Belinda Phipson

References

Phipson, B., and Oshlack, A. (2014). A method for detecting differential variability in methylation data shows CpG islands are highly variably methylated in cancers. *Genome Biology*, **15**:465.

See Also

`varFit`

Examples

```
# Randomly generate data for a 2 group problem with 100 CpG sites and 5
# arrays in each group
y <- matrix(rnorm(1000),ncol=10)

group <- factor(rep(c(1,2),each=5))
design <- model.matrix(~group)

# Get absolute Levene Residuals
resid <- getLeveneResiduals(y,design)

# Plot the first CpG
barplot(resid$data[1,],col=rep(c(2,4),each=5),
ylab="Absolute Levene Residuals",names=group)
```

`getMappedEntrezIDs` *Get mapped Entrez Gene IDs from CpG probe names*

Description

Given a set of CpG probe names and optionally all the CpG sites tested, this function outputs a list containing the mapped Entrez Gene IDs as well as the numbers of probes per gene, and a vector indicating significance.

Usage

```
getMappedEntrezIDs(
  sig.cpg,
  all.cpg = NULL,
  array.type = c("450K", "EPIC"),
  anno = NULL,
 genomic.features = c("ALL", "TSS200", "TSS1500", "Body", "1stExon", "3'UTR", "5'UTR",
    "ExonBnd")
)
```

Arguments

`sig.cpg`	Character vector of significant CpG sites used for testing gene set enrichment.
`all.cpg`	Character vector of all CpG sites tested. Defaults to all CpG sites on the array.
`array.type`	The Illumina methylation array used. Options are "450K" or "EPIC".
`anno`	Optional. A `DataFrame` object containing the complete array annotation as generated by the `minfi getAnnotation` function. Speeds up execution, if provided.
`genomic.features`	Character vector or scalar indicating whether the gene set enrichment analysis should be restricted to CpGs from specific genomic locations. Options are "ALL", "TSS200","TSS1500","Body","1stExon", "3'UTR","5'UTR","ExonBnd"; and the user can select any combination. Defaults to "ALL".

Details

This function is used by the gene set testing functions `gometh` and `gsameth`. It maps the significant CpG probe names to Entrez Gene IDs, as well as all the CpG sites tested. It also calculates the numbers of probes for gene. Input CpGs are able to be restricted by genomic features using the `genomic.features` argument.

Genes associated with each CpG site are obtained from the annotation package `IlluminaHumanMethylation450kanno.ilmn` if the array type is "450K". For the EPIC array, the annotation package `IlluminaHumanMethylationEPICanno.ilm10b4.hg` is used. To use a different annotation package, please supply it using the `anno` argument.

Value

A list with the following elements

sig.eg	mapped Entrez Gene IDs for the significant probes
universe	mapped Entrez Gene IDs for all probes on the array, or for all the CpG probes tested.
freq	table output with numbers of probes associated with each gene
equiv	table output with equivalent numbers of probes associated with each gene taking into account multi-gene bias
de	a vector of ones and zeroes of the same length of universe indicating which genes in the universe are significantly differentially methylated.
fract.counts	a dataframe with 2 columns corresponding to the Entrez Gene IDS for the significant probes and the associated weight to account for multi-gene probes.

Author(s)

Belinda Phipson

See Also

gometh,gsameth

Examples

```
## Not run:  # to avoid timeout on Bioconductor build
library(IlluminaHumanMethylation450kanno.ilmn12.hg19)
library(org.Hs.eg.db)
library(limma)
ann <- getAnnotation(IlluminaHumanMethylation450kanno.ilmn12.hg19)

# Randomly select 1000 CpGs to be significantly differentially methylated
sigcpgs <- sample(rownames(ann),1000,replace=FALSE)

# All CpG sites tested
allcpgs <- rownames(ann)

mappedEz <- getMappedEntrezIDs(sigcpgs,allcpgs,array.type="450K")
names(mappedEz)
# Entrez IDs of the significant genes
mappedEz$sig.eg[1:10]
# Entrez IDs for the universe
mappedEz$universe[1:10]
# Number of CpGs per gene
mappedEz$freq[1:10]
# Equivalent numbers of CpGs measured per gene
mappedEz$equiv[1:10]
A vector of 0s and 1s indicating which genes in the universe are significant
mappedEz$de[1:10]

## End(Not run)
```

gometh *Gene ontology testing for Ilumina methylation array data*

Description

Tests gene ontology enrichment for significant CpGs from Illumina's Infinium HumanMethylation450 or MethylationEPIC array, taking into account two different sources of bias: 1) the differing number of probes per gene present on the array, and 2) CpGs that are annotated to multiple genes.

Usage

```
gometh(
  sig.cpg,
  all.cpg = NULL,
  collection = c("GO", "KEGG"),
  array.type = c("450K", "EPIC"),
  plot.bias = FALSE,
  prior.prob = TRUE,
  anno = NULL,
  equiv.cpg = TRUE,
  fract.counts = TRUE,
 genomic.features = c("ALL", "TSS200", "TSS1500", "Body", "1stExon", "3'UTR", "5'UTR",
    "ExonBnd"),
  sig.genes = FALSE
)
```

Arguments

sig.cpg	Character vector of significant CpG sites to test for GO term enrichment.
all.cpg	Character vector of all CpG sites tested. Defaults to all CpG sites on the array.
collection	The collection of pathways to test. Options are "GO" and "KEGG". Defaults to "GO".
array.type	The Illumina methylation array used. Options are "450K" or "EPIC". Defaults to "450K".
plot.bias	Logical, if true a plot showing the bias due to the differing numbers of probes per gene will be displayed.
prior.prob	Logical, if true will take into account the probability of significant differential methylation due to numbers of probes per gene. If false, a hypergeometric test is performed ignoring any bias in the data.
anno	Optional. A DataFrame object containing the complete array annotation as generated by the minfi getAnnotation function. Speeds up execution, if provided.
equiv.cpg	Logical, if true then equivalent numbers of cpgs are used for odds calculation rather than total number cpgs. Only used if prior.prob=TRUE.

`fract.counts` Logical, if true then fractional counting of Cpgs is used to account for CpGs that are annotated to multiple genes. Only used if `prior.prob=TRUE`.

`genomic.features` Character vector or scalar indicating whether the gene set enrichment analysis should be restricted to CpGs from specific genomic locations. Options are "ALL", "TSS200","TSS1500","Body","1stExon", "3'UTR","5'UTR","ExonBnd"; and the user can select any combination. Defaults to "ALL".

`sig.genes` Logical, if true then the significant differentially methylated genes that overlap with the gene set of interest is outputted as the final column in the results table. Default is FALSE.

Details

This function takes a character vector of significant CpG sites, maps the CpG sites to Entrez Gene IDs, and tests for GO term or KEGG pathway enrichment using a Wallenius' non central hypergeometric test, taking into account the number of CpG sites per gene on the 450K/EPIC array and multi-gene annotated CpGs. Geeleher et al. (2013) showed that a severe bias exists when performing gene set analysis for genome-wide methylation data that occurs due to the differing numbers of CpG sites profiled for each gene. `gometh` is based on the `goseq` method (Young et al., 2010), and is a modification of the `goana` function in the `limma` package. If `prior.prob` is set to FALSE, then prior probabilities are not used and it is assumed that each gene is equally likely to have a significant CpG site associated with it.

The testing now also takes into account that some CpGs are annotated to multiple genes. For a small number of gene families, this previously caused their associated GO categories/gene sets to be erroneously overrepresented and thus highly significant. If `fract.counts=FALSE` then CpGs are allowed to map to multiple genes (this is NOT recommended).

A new feature of `gometh` and `gsameth` is the ability to restrict the input CpGs by genomic feature with the argument `genomic.features`. The possible options include "ALL", "TSS200", "TSS1500", "Body", "1stExon", "3'UTR", "5'UTR" and "ExonBnd", and the user may specify any combination. Please not that "ExonBnd" is not an annotated feature on 450K arrays. For example if you are interested in the promoter region only, you could specify `genomic.features = c("TSS1500","TSS200","1stExon")`. The default behaviour is to test all input CpGs `sig.cpg` even if the user specifies "ALL" and one or more other features.

Genes associated with each CpG site are obtained from the annotation package `IlluminaHumanMethylation450kanno.ilmn` if the array type is "450K". For the EPIC array, the annotation package `IlluminaHumanMethylationEPICanno.ilm10b4.hg` is used. To use a different annotation package, please supply it using the `anno` argument.

If you are interested in which genes overlap with the genes in the gene set, setting `sig.genes` to TRUE will output an additional column in the results data frame that contains all the significant differentially methylated gene symbols, comma separated. The default is FALSE.

In order to get a list which contains the mapped Entrez gene IDs, please use the `getMappedEntrezIDs` function. `gometh` tests all GO or KEGG terms, and false discovery rates are calculated using the method of Benjamini and Hochberg (1995). The `topGSA` function can be used to display the top 20 most enriched pathways.

For more generalised gene set testing where the user can specify the gene set/s of interest to be tested, please use the `gsameth` function. If you are interested in performing gene set testing following a region analysis, then the functions `goregion` and `gsaregion` can be used.

Value

A data frame with a row for each GO or KEGG term and the following columns:

Term	GO term if testing GO pathways
Ont	ontology that the GO term belongs to if testing GO pathways. "BP" - biological process, "CC" - cellular component, "MF" - molecular function.
Pathway	the KEGG pathway being tested if testing KEGG terms.
N	number of genes in the GO or KEGG term
DE	number of genes that are differentially methylated
P.DE	p-value for over-representation of the GO or KEGG term term
FDR	False discovery rate
SigGenesInSet	Significant differentially methylated genes overlapping with the gene set of interest.

Author(s)

Belinda Phipson

References

Phipson, B., Maksimovic, J., and Oshlack, A. (2016). missMethyl: an R package for analysing methylation data from Illuminas HumanMethylation450 platform. *Bioinformatics*, **15**;32(2), 286–8.

Geeleher, P., Hartnett, L., Egan, L. J., Golden, A., Ali, R. A. R., and Seoighe, C. (2013). Gene-set analysis is severely biased when applied to genome-wide methylation data. *Bioinformatics*, **29**(15), 1851–1857.

Young, M. D., Wakefield, M. J., Smyth, G. K., and Oshlack, A. (2010). Gene ontology analysis for RNA-seq: accounting for selection bias. *Genome Biology*, 11, R14.

Ritchie, M. E., Phipson, B., Wu, D., Hu, Y., Law, C. W., Shi, W., and Smyth, G. K. (2015). limma powers differential expression analyses for RNA-sequencing and microarray studies. *Nucleic Acids Research*, gkv007.

Benjamini, Y., and Hochberg, Y. (1995). Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society Series*, B, **57**, 289-300.

See Also

gsameth,goregion,gsaregion,getMappedEntrezIDs

Examples

```
## Not run:  # to avoid timeout on Bioconductor build
library(IlluminaHumanMethylation450kanno.ilmn12.hg19)
library(limma)
ann <- getAnnotation(IlluminaHumanMethylation450kanno.ilmn12.hg19)
# Randomly select 1000 CpGs to be significantly differentially methylated
sigcpgs <- sample(rownames(ann),1000,replace=FALSE)
```

```
# All CpG sites tested
allcpgs <- rownames(ann)

# GO testing with prior probabilities taken into account
# Plot of bias due to differing numbers of CpG sites per gene
gst <- gometh(sig.cpg = sigcpgs, all.cpg = allcpgs, collection = "GO",
              plot.bias = TRUE, prior.prob = TRUE, anno = ann)
# Total number of GO categories significant at 5% FDR
table(gst$FDR<0.05)
# Table of top GO results
topGSA(gst)

# GO testing ignoring bias
gst.bias <- gometh(sig.cpg = sigcpgs, all.cpg = allcpgs, collection = "GO",
                   prior.prob=FALSE, anno = ann)
# Total number of GO categories significant at 5% FDR ignoring bias
table(gst.bias$FDR<0.05)
# Table of top GO results ignoring bias
topGSA(gst.bias)

# GO testing ignoring multi-mapping CpGs
gst.multi <- gometh(sig.cpg = sigcpgs, all.cpg = allcpgs, collection = "GO",
              plot.bias = TRUE, prior.prob = TRUE, fract.counts = FALSE,
              anno = ann)
topGSA(gst.multi, n=10)

# Restrict to CpGs in promoter regions
gst.promoter <- gometh(sig.cpg = sigcpgs, all.cpg = allcpgs,
                collection = "GO", anno = ann,
                genomic.features=c("TSS200","TSS1500","1stExon"))
topGSA(gst.promoter)

# KEGG testing
kegg <- gometh(sig.cpg = sigcpgs, all.cpg = allcpgs, collection = "KEGG",
                prior.prob=TRUE, anno = ann)
# Table of top KEGG results
topGSA(kegg)

# Add significant genes to KEGG output
kegg.siggenes <- gometh(sig.cpg = sigcpgs, all.cpg = allcpgs,
                        collection = "KEGG", anno = ann, sig.genes = TRUE)
# Output top 5 KEGG pathways
topGSA(kegg.siggenes, n=5)

## End(Not run)
```

---

goregion *Gene ontology testing of DMRs for Ilumina methylation array data*

Description

Tests gene ontology or KEGG pathway enrichment for differentially methylated regions (DMRs) identified from Illumina's Infinium HumanMethylation450 or MethylationEPIC array, taking into account the differing number of probes per gene present on the array.

Usage

```
goregion(
  regions,
  all.cpg = NULL,
  collection = c("GO", "KEGG"),
  array.type = c("450K", "EPIC"),
  plot.bias = FALSE,
  prior.prob = TRUE,
  anno = NULL,
  equiv.cpg = TRUE,
  fract.counts = TRUE,
 genomic.features = c("ALL", "TSS200", "TSS1500", "Body", "1stExon", "3'UTR", "5'UTR",
    "ExonBnd"),
  sig.genes = FALSE
)
```

Arguments

`regions`	`GRanges` object of DMR coordinates to test for GO term enrichment.
`all.cpg`	Character vector of all CpG sites tested. Defaults to all CpG sites on the array.
`collection`	The collection of pathways to test. Options are "GO" and "KEGG". Defaults to "GO".
`array.type`	The Illumina methylation array used. Options are "450K" or "EPIC". Defaults to "450K".
`plot.bias`	Logical, if true a plot showing the bias due to the differing numbers of probes per gene will be displayed.
`prior.prob`	Logical, if true will take into account the probability of significant differentially methylation due to numbers of probes per gene. If false, a hypergeometric test is performed ignoring any bias in the data.
`anno`	Optional. A `DataFrame` object containing the complete array annotation as generated by the `minfi`. `getAnnotation` function. Speeds up execution, if provided.
`equiv.cpg`	Logical, if true then equivalent numbers of cpgs are used for odds calculation rather than total number cpgs. Only used if `prior.prob=TRUE`.
`fract.counts`	Logical, if true then fractional counting of cpgs is used to account for cgps that map to multiple genes. Only used if `prior.prob=TRUE`.
`genomic.features`	Character vector or scalar indicating whether the gene set enrichment analysis should be restricted to CpGs from specific genomic locations. Options are "ALL", "TSS200","TSS1500","Body","1stExon", "3'UTR","5'UTR","ExonBnd"; and the user can select any combination. Defaults to "ALL".

`sig.genes` Logical, if true then the significant differentially methylated genes that overlap with the gene set of interest is outputted as the final column in the results table. Default is FALSE.

Details

This function takes a `GRanges` object of DMR coordinates, maps them to CpG sites on the array and then to Entrez Gene IDs, and tests for GO term or KEGG pathway enrichment using Wallenius' noncentral hypergeometric test, taking into account the number of CpG sites per gene on the 450K/EPIC array. If `prior.prob` is set to FALSE, then prior probabilities are not used and it is assumed that each gene is equally likely to have a significant CpG site associated with it. Please not that we have tested `goregion` and `gsaregion` extensively using the `DMRCate` package to identify differentially methylated regions (Peters, et al., 2015).

The testing now also takes into account that some CpGs map to multiple genes. For a small number of gene families, this previously caused their associated GO categories/gene sets to be erroneously overrepresented and thus highly significant. If `fract.counts=FALSE` then CpGs are allowed to map to multiple genes (this is NOT recommended).

Genes associated with each CpG site are obtained from the annotation package `IlluminaHumanMethylation450kanno.ilmn` if the array type is "450K". For the EPIC array, the annotation package `IlluminaHumanMethylationEPICanno.ilm10b4.hg` is used. To use a different annotation package, please supply it using the `anno` argument.

In order to get a list which contains the mapped Entrez gene IDS, please use the `getMappedEntrezIDs` function. `goregion` tests all GO or KEGG terms, and false discovery rates are calculated using the method of Benjamini and Hochberg (1995). The `topGSA` function can be used to display the top 20 most enriched pathways.

If you are interested in which genes overlap with the genes in the gene set, setting `sig.genes` to TRUE will output an additional column in the results data frame that contains all the significant differentially methylated gene symbols, comma separated. The default is FALSE.

For more generalised gene set testing where the user can specify the gene set/s of interest to be tested, please use the `gsaregion` function.

Value

A data frame with a row for each GO or KEGG term and the following columns:

`Term`	GO term if testing GO pathways
`Ont`	ontology that the GO term belongs to if testing GO pathways. "BP" - biological process, "CC" - cellular component, "MF" - molecular function.
`Pathway`	the KEGG pathway being tested if testing KEGG terms.
`N`	number of genes in the GO or KEGG term
`DE`	number of genes that are differentially methylated
`P.DE`	p-value for over-representation of the GO or KEGG term term
`FDR`	False discovery rate
`SigGenesInSet`	Significant differentially methylated genes overlapping with the gene set of interest.

Author(s)

Jovana Maksimovic

References

Phipson, B., Maksimovic, J., and Oshlack, A. (2016). missMethyl: an R package for analysing methylation data from Illuminas HumanMethylation450 platform. *Bioinformatics*, **15**;32(2), 286–8.

Geeleher, P., Hartnett, L., Egan, L. J., Golden, A., Ali, R. A. R., and Seoighe, C. (2013). Gene-set analysis is severely biased when applied to genome-wide methylation data. *Bioinformatics*, **29**(15), 1851–1857.

Young, M. D., Wakefield, M. J., Smyth, G. K., and Oshlack, A. (2010). Gene ontology analysis for RNA-seq: accounting for selection bias. *Genome Biology*, 11, R14.

Ritchie, M. E., Phipson, B., Wu, D., Hu, Y., Law, C. W., Shi, W., and Smyth, G. K. (2015). limma powers differential expression analyses for RNA-sequencing and microarray studies. *Nucleic Acids Research*, gkv007.

Benjamini, Y., and Hochberg, Y. (1995). Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society Series*, B, **57**, 289-300.

Peters, T.J., Buckley, M.J., Statham, A.L., Pidsley, R., Samaras, K., Lord, R.V., Clark, S.J.,Molloy, P.L. (2015). De novo identification of differentially methylated regions in the human genome. *Epigenetics & Chromatin*, **8**, 6.

See Also

gometh,gsameth,gsaregion

Examples

```
## Not run:  # to avoid timeout on Bioconductor build
library(IlluminaHumanMethylationEPICanno.ilm10b4.hg19)
library(limma)
library(DMRcate)
library(ExperimentHub)

# Follow the example for the dmrcate function to get some EPIC data from
# ExperimentHub
eh <- ExperimentHub()
FlowSorted.Blood.EPIC <- eh[["EH1136"]]
tcell <- FlowSorted.Blood.EPIC[,colData(FlowSorted.Blood.EPIC)$CD4T==100 |
                               colData(FlowSorted.Blood.EPIC)$CD8T==100]
detP <- detectionP(tcell)
remove <- apply(detP, 1, function (x) any(x > 0.01))
tcell <- tcell[!remove,]
tcell <- preprocessFunnorm(tcell)
#Subset to chr2 only
tcell <- tcell[seqnames(tcell) == "chr2",]
tcellms <- getM(tcell)
tcellms.noSNPs <- rmSNPandCH(tcellms, dist=2, mafcut=0.05)
tcell$Replicate[tcell$Replicate==""] <- tcell$Sample_Name[tcell$Replicate==""]
```

```
tcellms.noSNPs <- avearrays(tcellms.noSNPs, tcell$Replicate)
tcell <- tcell[,!duplicated(tcell$Replicate)]
tcell <- tcell[rownames(tcellms.noSNPs),]
colnames(tcellms.noSNPs) <- colnames(tcell)
assays(tcell)[["M"]] <- tcellms.noSNPs
assays(tcell)[["Beta"]] <- ilogit2(tcellms.noSNPs)

# Perform region analysis
type <- factor(tcell$CellType)
design <- model.matrix(~type)
myannotation <- cpg.annotate("array", tcell, arraytype = "EPIC",
                             analysis.type="differential", design=design,
                             coef=2)
# Run DMRCate with beta value cut-off filter of 0.1
dmrcoutput <- dmrcate(myannotation, lambda=1000, C=2, betacutoff = 0.1)
regions <- extractRanges(dmrcoutput)
length(regions)

ann <- getAnnotation(IlluminaHumanMethylationEPICanno.ilm10b4.hg19)
# All CpG sites tested (limited to chr 2)
allcpgs <- rownames(tcell)
# GO testing with prior probabilities taken into account
# Plot of bias due to differing numbers of CpG sites per gene
gst <- goregion(regions = regions, all.cpg = allcpgs, collection = "GO",
                array.type = "EPIC", plot.bias = TRUE, prior.prob = TRUE,
                anno = ann)
# Table of top GO results
topGSA(gst, n=10)

# KEGG testing
kegg <- goregion(regions = regions, all.cpg = allcpgs, collection = "KEGG",
                 array.type = "EPIC", prior.prob=TRUE, anno = ann)
# Table of top KEGG results
topGSA(kegg, n=10)

# Restrict to promoter regions
gst.prom <- goregion(regions = regions, all.cpg = allcpgs, collection = "GO",
                array.type = "EPIC", plot.bias = TRUE, prior.prob = TRUE,
                anno = ann, genomic.features = c("TSS200","TSS1500"))
topGSA(gst.prom, n=10)

# Add significant genes in gene set to KEGG output
kegg <- goregion(regions = regions, all.cpg = allcpgs, collection = "KEGG",
                 array.type = "EPIC", prior.prob=TRUE, anno = ann,
                 sig.genes = TRUE)
# Table of top KEGG results
topGSA(kegg, n=5)

## End(Not run)
```

gsameth *Generalised gene set testing for Illumina's methylation array data*

Description

Given a user specified list of gene sets to test, gsameth tests whether significantly differentially methylated CpG sites are enriched in these gene sets.

Usage

```
gsameth(
  sig.cpg,
  all.cpg = NULL,
  collection,
  array.type = c("450K", "EPIC"),
  plot.bias = FALSE,
  prior.prob = TRUE,
  anno = NULL,
  equiv.cpg = TRUE,
  fract.counts = TRUE,
 genomic.features = c("ALL", "TSS200", "TSS1500", "Body", "1stExon", "3'UTR", "5'UTR",
    "ExonBnd"),
  sig.genes = FALSE
)
```

Arguments

sig.cpg	Character vector of significant CpG sites to test for gene set enrichment.
all.cpg	Character vector of all CpG sites tested. Defaults to all CpG sites on the array.
collection	A list of user specified gene sets to test. Can also be a single character vector gene set. Gene identifiers must be Entrez Gene IDs.
array.type	The Illumina methylation array used. Options are "450K" or "EPIC". Defaults to "450K".
plot.bias	Logical, if true a plot showing the bias due to the differing numbers of probes per gene will be displayed
prior.prob	Logical, if true will take into account the probability of significant differentially methylation due to numbers of probes per gene. If false, a hypergeometric test is performed ignoring any bias in the data.
anno	Optional. A DataFrame object containing the complete array annotation as generated by the minfi getAnnotation function. Speeds up execution, if provided.
equiv.cpg	Logical, if true then equivalent numbers of cpgs are used for odds calculation rather than total number cpgs. Only used if prior.prob=TRUE.
fract.counts	Logical, if true then fractional counting of cpgs is used to account for cgps that map to multiple genes. Only used if prior.prob=TRUE.

genomic.features
: Character vector or scalar indicating whether the gene set enrichment analysis should be restricted to CpGs from specific genomic locations. Options are "ALL", "TSS200","TSS1500","Body","1stExon", "3'UTR","5'UTR","ExonBnd"; and the user can select any combination. Defaults to "ALL".

sig.genes
: Logical, if true then the significant differentially methylated genes that overlap with the gene set of interest is outputted as the final column in the results table. Default is FALSE.

Details

This function extends gometh, which only tests GO and KEGG pathways. gsameth can take a list of user specified gene sets and test whether the significant CpG sites are enriched in these pathways. gsameth maps the CpG sites to Entrez Gene IDs and tests for pathway enrichment using Wallenius' concentral hypergeometric test, taking into account the number of CpG sites per gene on the 450K/EPIC arrays. Please note the gene ids for the collection of gene,sets must be Entrez Gene IDs. If prior.prob is set to FALSE, then prior probabilities are not used and it is assumed that each gene is equally likely to have a significant CpG site associated with it.

The testing now also takes into account that some CpGs map to multiple genes. For a small number of gene families, this previously caused their associated GO categories/gene sets to be erroneously overrepresented and thus highly significant. If fract.counts=FALSE then CpGs are allowed to map to multiple genes (this is NOT recommended).

A new feature of gometh and gsameth is the ability to restrict the input CpGs by genomic feature with the argument genomic.features. The possible options include "ALL", "TSS200", "TSS1500", "Body", "1stExon", "3'UTR", "5'UTR" and "ExonBnd", and the user may specify any combination. Please not that "ExonBnd" is not an annotatedfeature on 450K arrays. For example if you are interested in the promoter region only, you could specify genomic.features = c("TSS1500","TSS200","1stExon"). The default behaviour is to test all input CpGs sig.cpg even if the user specifies "ALL" and one or more other features.

Genes associated with each CpG site are obtained from the annotation package IlluminaHumanMethylation450kanno.ilmn if the array type is "450K". For the EPIC array, the annotation package IlluminaHumanMethylationEPICanno.ilm10b4.hg is used. To use a different annotation package, please supply it using the anno argument.

In order to get a list which contains the mapped Entrez gene IDS, please use the getMappedEntrezIDs function.

If you are interested in which genes overlap with the genes in the gene set, setting sig.genes to TRUE will output an additional column in the results data frame that contains all the significant differentially methylated gene symbols, comma separated. The default is FALSE.

Value

A data frame with a row for each gene set and the following columns:

N
: number of genes in the gene set

DE
: number of genes that are differentially methylated

P.DE
: p-value for over-representation of the gene set

FDR
: False discovery rate, calculated using the method of Benjamini and Hochberg (1995).

`SigGenesInSet` Significant differentially methylated genes overlapping with the gene set of interest.

Author(s)

Belinda Phipson

References

Phipson, B., Maksimovic, J., and Oshlack, A. (2016). missMethyl: an R package for analysing methylation data from Illuminas HumanMethylation450 platform. *Bioinformatics*, **15**;32(2), 286–8.

Geeleher, P., Hartnett, L., Egan, L. J., Golden, A., Ali, R. A. R., and Seoighe, C. (2013). Gene-set analysis is severely biased when applied to genome-wide methylation data. *Bioinformatics*, **29**(15), 1851–1857.

Young, M. D., Wakefield, M. J., Smyth, G. K., and Oshlack, A. (2010). Gene ontology analysis for RNA-seq: accounting for selection bias. *Genome Biology*, 11, R14.

Ritchie, M. E., Phipson, B., Wu, D., Hu, Y., Law, C. W., Shi, W., and Smyth, G. K. (2015). limma powers differential expression analyses for RNA-sequencing and microarray studies. *Nucleic Acids Research*, gkv007.

Benjamini, Y., and Hochberg, Y. (1995). Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society Series*, B, **57**, 289-300.

See Also

`gometh`, `getMappedEntrezIDs`

Examples

```
## Not run:  # to avoid timeout on Bioconductor build
library(IlluminaHumanMethylation450kanno.ilmn12.hg19)
library(org.Hs.eg.db)
library(limma)
ann <- getAnnotation(IlluminaHumanMethylation450kanno.ilmn12.hg19)
# Randomly select 1000 CpGs to be significantly differentially methylated
sigcpgs <- sample(rownames(ann),1000,replace=FALSE)
# All CpG sites tested
allcpgs <- rownames(ann)
# Use org.Hs.eg.db to extract a GO term
GOtoID <- suppressMessages(select(org.Hs.eg.db, keys=keys(org.Hs.eg.db),
                                  columns=c("ENTREZID","GO"),
                                  keytype="ENTREZID"))
setname1 <- GOtoID$GO[1]
setname1
keep.set1 <- GOtoID$GO %in% setname1
set1 <- GOtoID$ENTREZID[keep.set1]
setname2 <- GOtoID$GO[2]
setname2
keep.set2 <- GOtoID$GO %in% setname2
set2 <- GOtoID$ENTREZID[keep.set2]
```

```
# Make the gene sets into a list
sets <- list(set1, set2)
names(sets) <- c(setname1,setname2)
# Testing with prior probabilities taken into account
# Plot of bias due to differing numbers of CpG sites per gene
gst <- gsameth(sig.cpg = sigcpgs, all.cpg = allcpgs, collection = sets,
               plot.bias = TRUE, prior.prob = TRUE)
topGSA(gst)

# Add significant gene symbols in each set to output
gst <- gsameth(sig.cpg = sigcpgs, all.cpg = allcpgs, collection = sets,
               plot.bias = TRUE, prior.prob = TRUE, sig.genes = TRUE)
topGSA(gst)

# Testing ignoring bias
gst.bias <- gsameth(sig.cpg = sigcpgs, all.cpg = allcpgs, collection = sets,
                    prior.prob = FALSE)
topGSA(gst.bias)

# Restrict to CpGs in gene bodies
gst.body <- gsameth(sig.cpg = sigcpgs, all.cpg = allcpgs, collection = sets,
                    genomic.features = "Body")
topGSA(gst.body)


## End(Not run)
```

`gsaregion` *Generalised gene set testing for Illumina's methylation array data*

Description

Given a user specified list of gene sets to test, `gsaregion` tests whether differentially methylated regions (DMRs) identified from Illumina's Infinium HumanMethylation450 or MethylationEPIC array are enriched, taking into account the differing number of probes per gene present on the array.

Usage

```
gsaregion(
  regions,
  all.cpg = NULL,
  collection,
  array.type = c("450K", "EPIC"),
  plot.bias = FALSE,
  prior.prob = TRUE,
  anno = NULL,
  equiv.cpg = TRUE,
  fract.counts = TRUE,
```

```
  genomic.features = c("ALL", "TSS200", "TSS1500", "Body", "1stExon", "3'UTR", "5'UTR",
     "ExonBnd"),
  sig.genes = FALSE
)
```

Arguments

regions	GRanges Object of DMR coordinates to test for GO term enrichment.
all.cpg	Character vector of all CpG sites tested. Defaults to all CpG sites on the array.
collection	A list of user specified gene sets to test. Can also be a single character vector gene set. Gene identifiers must be Entrez Gene IDs.
array.type	The Illumina methylation array used. Options are "450K" or "EPIC". Defaults to "450K".
plot.bias	Logical, if true a plot showing the bias due to the differing numbers of probes per gene will be displayed.
prior.prob	Logical, if true will take into account the probability of significant differentially methylation due to numbers of probes per gene. If false, a hypergeometric test is performed ignoring any bias in the data.
anno	Optional. A DataFrame object containing the complete array annotation as generated by the minfi getAnnotation function. Speeds up execution, if provided.
equiv.cpg	Logical, if true then equivalent numbers of cpgs are used for odds calculation rather than total number cpgs. Only used if prior.prob=TRUE.
fract.counts	Logical, if true then fractional counting of cpgs is used to account for cgps that map to multiple genes. Only used if prior.prob=TRUE.
genomic.features	Character vector or scalar indicating whether the gene set enrichment analysis should be restricted to CpGs from specific genomic locations. Options are "ALL", "TSS200","TSS1500","Body","1stExon", "3'UTR","5'UTR","ExonBnd"; and the user can select any combination. Defaults to "ALL".
sig.genes	Logical, if true then the significant differentially methylated genes that overlap with the gene set of interest is outputted as the final column in the results table. Default is FALSE.

Details

This function extends goregion, which only tests GO and KEGG pathways. gsaregion can take a list of user specified gene sets and test whether the significant DMRs are enriched in these pathways. This function takes a GRanges object of DMR coordinates, maps them to CpG sites on the array and then to Entrez Gene IDs, and tests for enrichment using Wallenius' noncentral hypergeometric test, taking into account the number of CpG sites per gene on the 450K/EPIC array. If prior.prob is set to FALSE, then prior probabilities are not used and it is assumed that each gene is equally likely to have a significant CpG site associated with it.

The testing now also takes into account that some CpGs map to multiple genes. For a small number of gene families, this previously caused their associated GO categories/gene sets to be erroneously overrepresented and thus highly significant. If fract.counts=FALSE then CpGs are allowed to map to multiple genes (this is NOT recommended).

Genes associated with each CpG site are obtained from the annotation package `IlluminaHumanMethylation450kanno.ilmn` if the array type is "450K". For the EPIC array, the annotation package `IlluminaHumanMethylationEPICanno.ilm10b4.hg` is used. To use a different annotation package, please supply it using the `anno` argument.

In order to get a list which contains the mapped Entrez gene IDS, please use the `getMappedEntrezIDs` function. The `topGSA` function can be used to display the top 20 most enriched pathways.

If you are interested in which genes overlap with the genes in the gene set, setting `sig.genes` to TRUE will output an additional column in the results data frame that contains all the significant differentially methylated gene symbols, comma separated. The default is FALSE.

Value

A data frame with a row for each gene set and the following columns:

`N`	number of genes in the gene set
`DE`	number of genes that are differentially methylated
`P.DE`	p-value for over-representation of the gene set
`FDR`	False discovery rate, calculated using the method of Benjamini and Hochberg (1995).
`SigGenesInSet`	Significant differentially methylated genes overlapping with the gene set of interest.

Author(s)

Jovana Maksimovic

References

Phipson, B., Maksimovic, J., and Oshlack, A. (2016). missMethyl: an R package for analysing methylation data from Illuminas HumanMethylation450 platform. *Bioinformatics*, **15**;32(2), 286–8.

Geeleher, P., Hartnett, L., Egan, L. J., Golden, A., Ali, R. A. R., and Seoighe, C. (2013). Gene-set analysis is severely biased when applied to genome-wide methylation data. *Bioinformatics*, **29**(15), 1851–1857.

Young, M. D., Wakefield, M. J., Smyth, G. K., and Oshlack, A. (2010). Gene ontology analysis for RNA-seq: accounting for selection bias. *Genome Biology*, 11, R14.

Ritchie, M. E., Phipson, B., Wu, D., Hu, Y., Law, C. W., Shi, W., and Smyth, G. K. (2015). limma powers differential expression analyses for RNA-sequencing and microarray studies. *Nucleic Acids Research*, gkv007.

Benjamini, Y., and Hochberg, Y. (1995). Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society Series*, B, **57**, 289-300.

See Also

`gometh`,`goregion`,`gsameth`,`getMappedEntrezIDs`

Examples

```
## Not run:  # to avoid timeout on Bioconductor build
library(IlluminaHumanMethylationEPICanno.ilm10b4.hg19)
library(limma)
library(DMRcate)
library(ExperimentHub)
library(org.Hs.eg.db)

# Follow the example for the dmrcate function to get some EPIC data from
# ExperimentHub
eh <- ExperimentHub()
FlowSorted.Blood.EPIC <- eh[["EH1136"]]
tcell <- FlowSorted.Blood.EPIC[,colData(FlowSorted.Blood.EPIC)$CD4T==100 |
                               colData(FlowSorted.Blood.EPIC)$CD8T==100]
detP <- detectionP(tcell)
remove <- apply(detP, 1, function (x) any(x > 0.01))
tcell <- tcell[!remove,]
tcell <- preprocessFunnorm(tcell)
#Subset to chr2 only
tcell <- tcell[seqnames(tcell) == "chr2",]
tcellms <- getM(tcell)
tcellms.noSNPs <- rmSNPandCH(tcellms, dist=2, mafcut=0.05)
tcell$Replicate[tcell$Replicate==""] <- tcell$Sample_Name[tcell$Replicate==""]
tcellms.noSNPs <- avearrays(tcellms.noSNPs, tcell$Replicate)
tcell <- tcell[,!duplicated(tcell$Replicate)]
tcell <- tcell[rownames(tcellms.noSNPs),]
colnames(tcellms.noSNPs) <- colnames(tcell)
assays(tcell)[["M"]] <- tcellms.noSNPs
assays(tcell)[["Beta"]] <- ilogit2(tcellms.noSNPs)

# Perform region analysis
type <- factor(tcell$CellType)
design <- model.matrix(~type)
myannotation <- cpg.annotate("array", tcell, arraytype = "EPIC",
                             analysis.type="differential", design=design,
                             coef=2)
# Run DMRCate with beta value cut-off filter of 0.1
dmrcoutput <- dmrcate(myannotation, lambda=1000, C=2, betacutoff = 0.1)
regions <- extractRanges(dmrcoutput)
length(regions)

ann <- getAnnotation(IlluminaHumanMethylationEPICanno.ilm10b4.hg19)
# All CpG sites tested (limited to chr 2)
allcpgs <- rownames(tcell)
# Use org.Hs.eg.db to extract a GO term
GOtoID <- suppressMessages(select(org.Hs.eg.db, keys=keys(org.Hs.eg.db),
                                  columns=c("ENTREZID","GO"),
                                  keytype="ENTREZID"))
keep.set1 <- GOtoID$GO %in% "GO:0010951"
set1 <- GOtoID$ENTREZID[keep.set1]
keep.set2 <- GOtoID$GO %in% "GO:0042742"
set2 <- GOtoID$ENTREZID[keep.set2]
```

```
keep.set3 <- GOtoID$GO %in% "GO:0031295"
set3 <- GOtoID$ENTREZID[keep.set3]
# Make the gene sets into a list
sets <- list(set1, set2, set3)
names(sets) <- c("GO:0010951","GO:0042742","GO:0031295")

# Testing with prior probabilities taken into account
# Plot of bias due to differing numbers of CpG sites per gene
gst <- gsaregion(regions = regions, all.cpg = allcpgs, collection = sets,
                 array.type = "EPIC", plot.bias = TRUE, prior.prob = TRUE,
                 anno = ann)
topGSA(gst)

# Add significant genes in gene set to output
gst <- gsaregion(regions = regions, all.cpg = allcpgs, collection = sets,
                 array.type = "EPIC", plot.bias = TRUE, prior.prob = TRUE,
                 anno = ann, sig.genes = TRUE)
topGSA(gst)

## End(Not run)
```

---

gsaseq *Generalised gene set testing for RNA-seq data*

Description

Given a user defined list of gene sets, gsaseq will test whether significantly differentially expressed genes are enriched in these gene sets.

Usage

```
gsaseq(
  sig.de,
  universe,
  collection,
  plot.bias = FALSE,
  gene.length = NULL,
  sort = TRUE
)
```

Arguments

sig.de	Character vector of significant differentially expressed genes to test for gene set enrichment. Must be Entrez Gene ID format.
universe	Character vector of all genes analysed in the experiment. Must be Entrez Gene ID format.
collection	A list of user specified gene sets to test. Can also be a single character vector gene set. Gene identifiers must be Entrez Gene IDs.

plot.bias	Logical, if true a plot showing gene length bias related to differential expression will be displayed.
gene.length	A vector containing the gene lengths for each gene in the same order as universe.
sort	Logical, if TRUE then the output dataframe is sorted by p-value.

Details

This function is a generalised version of goana and kegga from the limma package in that it can take a user-defined list of differentially expressed genes and perform gene set enrichment analysis, and is not limited to only testing GO and KEGG categories. It is not as flexible as goana and kegga. Please note the vector of differentially expressed genes and list of gene sets must be Entrez Gene IDs.

The gsaseq function will test for enrichment using a hypergeometric test if the gene.length parameter is NULL. If the gene.length parameter is supplied then the p-values are derived from Walllenius' noncentral hypergeometric distribution from the BiasedUrn package. Please note that the gene.length parameter must be in the same order and of the same length as the universe parameter.

Value

A data frame with a row for each gene set and the following columns:

N	number of genes in the gene set
DE	number of genes that are differentially expressed
P.DE	p-value for over-representation of the gene set
FDR	False discovery rate, calculated using the method of Benjamini and Hochberg (1995).

Author(s)

Belinda Phipson

See Also

goana,kegga,camera,roast

Examples

```
## Not run:  # to avoid timeout on Bioconductor build
library(org.Hs.eg.db)
# Use org.Hs.eg.db to extract GO terms
GOtoID <- suppressMessages(select(org.Hs.eg.db, keys=keys(org.Hs.eg.db),
                                  columns=c("ENTREZID","GO"),
                                  keytype="ENTREZID"))
head(GOtoID)

# Define the universe as random sample of 20000 genes in the annotation
universe <- sample(unique(GOtoID$ENTREZID),20000)
```

```
# Randomly sample 500 genes as DE
de.genes <- sample(universe, 500)

# Generate random gene lengths for genes in universe
# This is based on the true distribution of log(gene length) of genes in the
# hg19 genome
logGL <- rnorm(length(universe),mean=7.9, sd=1.154)
genelength <- exp(logGL)

# Define a list of gene sets containing two GO terms
setname1 <- GOtoID$GO[1]
setname1
keep.set1 <- GOtoID$GO %in% setname1
set1 <- GOtoID$ENTREZID[keep.set1]
setname2 <- GOtoID$GO[2]
setname2
keep.set2 <- GOtoID$GO %in% setname2
set2 <- GOtoID$ENTREZID[keep.set2]
# Make the gene sets into a list
sets <- list(set1, set2)
names(sets) <- c(setname1,setname2)

# Test for enrichment of gene sets with no gene length bias
# The genes are randomly selected so we don't expect significant results
gsaseq(sig.de = de.genes, universe = universe, collection = sets)

# Test for enrichment of gene sets taking into account gene length bias
# Since the gene lengths are randomly generated this shouldn't make much
# difference to the results
# Using log(gene length) or gene length doesn't make a difference to the
# p-values because the probability weighting function is transformation
# invariant
gsaseq(sig.de = de.genes, univers = universe, collection = sets,
gene.length = genelength)

## End(Not run)
```

RUVadj *RUV adjust*

Description

Post-process and summarize the results of call to RUVfit.

Usage

```
RUVadj(
  Y,
  fit,
```

```
  var.type = c("ebayes", "standard", "pooled"),
  p.type = c("standard", "rsvar", "evar"),
  cpginfo = NULL,
  ...
)
```

Arguments

Y	The original data matrix used in the call to RUVfit.
fit	A RUV model fit (a list) as returned by RUVfit.
var.type	Which type of estimate for sigma2 should be used from the call to variance_adjust? The options are "ebayes", "standard", or "pooled." See variance_adjust for details.
p.type	Which type of p-values should be used from the call to variance_adjust? The options are "standard", "rsvar", or "evar".
cpginfo	A matrix or dataframe containing information about the CpGs. This information is included in the summary that is returned.
...	Other parameters that can be passed to ruv function ruv_summary.

Details

This function post-processes the results of a call to RUVfit and then summarizes the output. The post-processing step primarily consists of a call to ruv_summary and variance_adjust, which computes various adjustments to variances, t-statistics, and and p-values. See variance_adjust for details. The var.type and p.type options determine which of these adjustments are used.

After post-processing, the results are summarized into a list containing 4 objects: 1) the data matrix Y; 2) a dataframe R containing information about the rows (samples); 3) a dataframe C containing information about the columns (features, e.g. genes), and 4) a list misc of other information returned by RUVfit.

Value

An list containing:

Y	The original data matrix..
R	A dataframe of sample-wise information, including X, Z, and any other data passed in with rowinfo.
C	A dataframe of cpg-wise information, including p-values, estimated regression coefficients, estimated variances, column means, an index of the negative controls, and any other data passed in with cpginfo.
misc	A list of additional information returned by RUVfit.

Author(s)

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

References

Benjamini, Y., and Hochberg, Y. (1995). Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society Series*, B, **57**, 289-300.

Gagnon-Bartsch JA, Speed TP. (2012). Using control genes to correct for unwanted variation in microarray data. *Biostatistics*. **13**(3), 539-52. Available at: http://biostatistics.oxfordjournals.org/content/13/3/539.full.

Gagnon-Bartsch, Jacob, and Speed. 2013. Removing Unwanted Variation from High Dimensional Data with Negative Controls. Available at: http://statistics.berkeley.edu/tech-reports/820.

Smyth, G. K. (2004). Linear models and empirical Bayes methods for assessing differential expression in microarray experiments. *Statistical Applications in Genetics and Molecular Biology*, Volume 3, Article 3. http://www.statsci.org/smyth/pubs/ebayes.pdf.

See Also

MArrayLM, RUV2, RUV4, RUVinv, RUVrinv, p.adjust, get_empirical_variances, sigmashrink

Examples

```
if(require(minfi) & require(minfiData) & require(limma)) {

# Get methylation data for a 2 group comparison
meth <- getMeth(MsetEx)
unmeth <- getUnmeth(MsetEx)
Mval <- log2((meth + 100)/(unmeth + 100))

group<-factor(pData(MsetEx)$Sample_Group)
design<-model.matrix(~group)

# Perform initial analysis to empirically identify negative control features
# when not known a priori
lFit <- lmFit(Mval,design)
lFit2 <- eBayes(lFit)
lTop <- topTable(lFit2,coef=2,num=Inf)

# The negative control features should *not* be associated with factor of
# interest but *should* be affected by unwanted variation
ctl <- rownames(Mval) %in% rownames(lTop[lTop$adj.P.Val > 0.5,])

# Perform RUV adjustment and fit
fit <- RUVfit(Y=Mval, X=group, ctl=ctl)
fit2 <- RUVadj(Y=Mval, fit=fit)

# Look at table of top results
top <- topRUV(fit2)
}
```

RUVfit *Remove unwanted variation when testing for differential methylation*

Description

Provides an interface similar to lmFit from limma to the RUV2, RUV4, RUVinv and RUVrinv functions from the ruv package, which facilitates the removal of unwanted variation in a differential methylation analysis. A set of negative control variables, as described in the references, must be specified.

Usage

```
RUVfit(
  Y,
  X,
  ctl,
  Z = 1,
  k = NULL,
  method = c("inv", "rinv", "ruv4", "ruv2"),
  ...
)
```

Arguments

Y	numeric matrix with rows corresponding to the features of interest such as CpG sites and columns corresponding to samples or arrays.
X	The factor(s) of interest. A m by p matrix, where m is the number of samples and p is the number of factors of interest. Very often p = 1. Factors and dataframes are also permissible, and converted to a matrix by design.matrix.
ctl	logical vector, length == nrow(Y). Features that are to be used as negative control variables are indicated as TRUE, all other features are FALSE.
Z	Any additional covariates to include in the model, typically a m by q matrix. Factors and dataframes are also permissible, and converted to a matrix by design.matrix. Alternatively, may simply be 1 (the default) for an intercept term. May also be NULL.
k	integer, required if method is "ruv2" or "ruv4". Indicates the number of unwanted factors to use. Can be 0.
method	character string, indicates which ruv method should be used.
...	additional arguments that can be passed to RUV2, RUV4, RUVinv and RUVrinv. See linked function documentation for details.

Details

This function depends on the ruv package and is used to estimate and adjust for unwanted variation in a differential methylation analysis. Briefly, the unwanted factors W are estimated using negative

control variables. Y is then regressed on the variables X, Z, and W. For methylation data, the analysis is performed on the M-values, defined as the log base 2 ratio of the methylated signal to the unmethylated signal.

Value

A list containing:

betahat	The estimated coefficients of the factor(s) of interest. A p by n matrix.
sigma2	Estimates of the features' variances. A vector of length n.
t	t statistics for the factor(s) of interest. A p by n matrix.
p	P-values for the factor(s) of interest. A p by n matrix.
Fstats	F statistics for testing all of the factors in X simultaneously..
Fpvals	P-values for testing all of the factors in X simultaneously.
multiplier	The constant by which sigma2 must be multiplied in order to get an estimate of the variance of betahat.
df	The number of residual degrees of freedom.
W	The estimated unwanted factors.
alpha	The estimated coefficients of W.
byx	The coefficients in a regression of Y on X (after both Y and X have been "adjusted" for Z). Useful for projection plots.
bwx	The coefficients in a regression of W on X (after X has been "adjusted" for Z). Useful for projection plots.
X	X. Included for reference.
k	k. Included for reference.
ctl	ctl. Included for reference.
Z	Z. Included for reference.
fullW0	Can be used to speed up future calls of RUVfit.
include.intercept	include.intercept. Included for reference.
method	Character variable with value indicating which RUV method was used. Included for reference.

Author(s)

Jovana Maksimovic

References

Gagnon-Bartsch JA, Speed TP. (2012). Using control genes to correct for unwanted variation in microarray data. *Biostatistics*. **13**(3), 539-52. Available at: http://biostatistics.oxfordjournals.org/content/13/3/539.full.

Gagnon-Bartsch, Jacob, and Speed. 2013. Removing Unwanted Variation from High Dimensional Data with Negative Controls. Available at: http://statistics.berkeley.edu/tech-reports/820.

See Also

RUV2, RUV4, RUVinv, RUVrinv, topRUV

Examples

```
if(require(minfi) & require(minfiData) & require(limma)) {
# Get methylation data for a 2 group comparison
meth <- getMeth(MsetEx)
unmeth <- getUnmeth(MsetEx)
Mval <- log2((meth + 100)/(unmeth + 100))
group <- factor(pData(MsetEx)$Sample_Group)
design <- model.matrix(~group)
# Perform initial analysis to empirically identify negative control features
# when not known a priori
lFit <- lmFit(Mval,design)
lFit2 <- eBayes(lFit)
lTop <- topTable(lFit2,coef=2,num=Inf)
# The negative control features should *not* be associated with factor of
# interest but *should* be affected by unwanted variation
ctl <- rownames(Mval) %in% rownames(lTop[lTop$adj.P.Val > 0.5,])
# Perform RUV adjustment and fit
fit <- RUVfit(Y=Mval, X=group, ctl=ctl)
fit2 <- RUVadj(Y=Mval, fit=fit)
# Look at table of top results
top <- topRUV(fit2)
}
```

SWAN *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

```
SWAN(data, verbose = FALSE)

## S3 method for class 'MethyLumiSet'
SWAN(data, verbose = FALSE)

## S3 method for class 'RGChannelSet'
SWAN(data, verbose = FALSE)

## Default S3 method:
SWAN(data, verbose = FALSE)
```

Arguments

data	An object of class either MethylSet, RGChannelSet or MethyLumiSet.
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.

NULL

NULL

NULL

Note

SWAN uses a random subset of probes to perform the within-array normalization. In order to achive reproducible results, the seed needs to be set using set.seed.

Author(s)

Jovana Maksimovic

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 MethyLumiSet and IlluminaMethylationManifest.

Examples

```
if (require(minfi) & require(minfiData)) {

  set.seed(100)
  datSwan1 <- SWAN(RGsetEx)
```

```
  dat <- preprocessRaw(RGsetEx)
  set.seed(100)
  datSwan2 <- SWAN(dat)

  head(getMeth(datSwan2)) == head(getMeth(datSwan1))
}
```

---

topGSA *Get table of top 20 enriched pathways*

Description

After using gsameth, calling topGSA will output the top 20 most significantly enriched pathways.

Usage

```
topGSA(gsa, number = 20, sort = TRUE)
```

Arguments

gsa	Matrix, from output of gsameth.
number	Scalar, number of pathway results to output. Default is 20.
sort	Logical, should the table be ordered by p-value. Default is TRUE.

Details

This function will output the top 20 most significant pathways from a pathway analysis using the gsameth function. The output is ordered by p-value.

Value

A matrix ordered by P.DE, with a row for each gene set and the following columns:

N	number of genes in the gene set
DE	number of genes that are differentially methylated
P.DE	p-value for over-representation of the gene set
FDR	False discovery rate, calculated using the method of Benjamini and Hochberg (1995)
.	
SigGenesInSet	Significant differentially methylated genes overlapping with the gene set of interest.

Author(s)

Belinda Phipson

See Also

gsameth

Examples

```
library(IlluminaHumanMethylation450kanno.ilmn12.hg19)
library(org.Hs.eg.db)
library(limma)
ann <- getAnnotation(IlluminaHumanMethylation450kanno.ilmn12.hg19)

# Randomly select 1000 CpGs to be significantly differentially methylated
sigcpgs <- sample(rownames(ann),1000,replace=FALSE)

# All CpG sites tested
allcpgs <- rownames(ann)

# Use org.Hs.eg.db to extract a GO term
GOtoID <- toTable(org.Hs.egGO2EG)
setname1 <- GOtoID$go_id[1]
setname1
keep.set1 <- GOtoID$go_id %in% setname1
set1 <- GOtoID$gene_id[keep.set1]
setname2 <- GOtoID$go_id[2]
setname2
keep.set2 <- GOtoID$go_id %in% setname2
set2 <- GOtoID$gene_id[keep.set2]

# Make the gene sets into a list
sets <- list(set1, set2)
names(sets) <- c(setname1,setname2)

# Testing with prior probabilities taken into account
# Plot of bias due to differing numbers of CpG sites per gene
gst <- gsameth(sig.cpg = sigcpgs, all.cpg = allcpgs, collection = sets,
plot.bias = TRUE, prior.prob = TRUE)
topGSA(gst)

# Testing ignoring bias
gst.bias <- gsameth(sig.cpg = sigcpgs, all.cpg = allcpgs, collection = sets,
                    prior.prob = FALSE)
topGSA(gst.bias)
```

---

topRUV *Table of top-ranked differentially methylated CpGs obatained from a differential methylation analysis using RUV*

Description

Extract a table of the top-ranked CpGs from a linear model fit after performing a differential methylation analysis using `RUVfit` and `RUVadj`.

Usage

```
topRUV(fitsum, number = 10, sort.by = c("p", "F.p"), p.BH = 1)
```

Arguments

fitsum	An object containing the summary fit object produced by RUVadj. The object should be a list.
number	integer, maximum number of genes to list. Default is 10.
sort.by	character string, what the results should be sorted by. Default is unadjusted p-value.
p.BH	numeric, cutoff value for Benjamini-Hochberg adjusted p-values. Only features with lower p-values are listed. Must be between 0 and 1. Default is 1.

Details

This function summarises the results of a differential methylation analysis performed using RUVfit, followed by RUVadj. The top ranked CpGs are sorted by p-value.

Value

Produces a dataframe with rows corresponding to the top number CpGs and the following columns: F.p F.p.BH p_X1 p.BH_X1 b_X1 sigma2 var.b_X1 fit.ctl mean

F.p	P-values for testing all of the factors of interest simultaneously.
F.p.BH	Benjamini-Hochberg adjusted p-values for testing all of the factors of interest simultaneously.
p_X1	p-values for the factor of interest.
p.BH_X1	Benjamini-Hochberg adjusted p-values for the factor of interest.
b_X1	The estimated coefficients of the factor of interest.
sigma2	Estimate of the methylation variance.
var.b_X1	Variance estimate of betahat.
fit.ctl	logical, indicating whether CpG was designated as a negative control.
mean	The mean methylation (M-value).

Author(s)

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

References

Benjamini, Y., and Hochberg, Y. (1995). Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society Series*, B, **57**, 289-300.

Smyth, G. K. (2004). Linear models and empirical Bayes methods for assessing differential expression in microarray experiments. *Statistical Applications in Genetics and Molecular Biology*, Volume 3, Article 3. http://www.statsci.org/smyth/pubs/ebayes.pdf.

See Also

RUVfit, RUVadj, MArrayLM

Examples

```
if(require(minfi) & require(minfiData) & require(limma)){

# Get methylation data for a 2 group comparison
meth <- getMeth(MsetEx)
unmeth <- getUnmeth(MsetEx)
Mval <- log2((meth + 100)/(unmeth + 100))

group <- factor(pData(MsetEx)$Sample_Group)
design <- model.matrix(~group)

# Perform initial analysis to empirically identify negative control features
# when *not* known a priori
lFit <- lmFit(Mval,design)
lFit2 <- eBayes(lFit)
lTop <- topTable(lFit2,coef=2,num=Inf)

# The negative control features should *not* be associated with factor of
# interest but *should* be affected by unwanted variation
ctl <- rownames(Mval) %in% rownames(lTop[lTop$adj.P.Val > 0.5,])

# Perform RUV adjustment and fit
fit <- RUVfit(Y=Mval, X=group, ctl=ctl)
fit2 <- RUVadj(Y=Mval, fit=fit)

# Look at table of top results
top <- topRUV(fit2)
}
```

---

topVar *Table of top-ranked differentially variable CpGs*

Description

Extract a table of the top-ranked CpGs from a linear model fit after a differential variability analysis.

Usage

```
topVar(fit, coef = NULL, number = 10, sort = TRUE)
```

Arguments

fit	List containing a linear model fit produced by varFit. The fit object should be of class MArrayLM.

coef	Column number or column name specifying which coefficient of the linear model fit is of interest. It should be the same coefficient that the differential variability testing was performed on. Default is last column of fit object.
number	Maximum number of genes to list. Default is 10.
sort	Logical, default is TRUE. Sorts output according the P-value. FALSE will return results in same order as fit object.

Details

This function summarises the results of a differential variability analysis performed with varFit. The p-values from the comparison of interest are adjusted using Benjamini and Hochberg's false discovery rate with the function p.adjust. The top ranked CpGs are selected by first ranking the adjusted p-values, then ranking the raw p-values. At this time no other sorting option is catered for.

Value

Produces a dataframe with rows corresponding to the top CpGs and the following columns:

genelist	one or more columns of annotation for each CpG, if the gene information is available in fit
AvgVar	average of the absolute or squared Levene residuals across all samples
DiffVar	estimate of the difference in the Levene residuals corresponding to the comparison of interest
t	moderated t-statistic
P.Value	raw p-value
Adj.P.Value	adjusted p-value

Author(s)

Belinda Phipson

References

Phipson, B., and Oshlack, A. (2014). A method for detecting differential variability in methylation data shows CpG islands are highly variably methylated in cancers. *Genome Biology*, **15**:465.

Benjamini, Y., and Hochberg, Y. (1995). Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society Series*, B, **57**, 289-300.

See Also

varFit, p.adjust

Examples

```
# Randomly generate data for a 2 group problem with 100 CpG sites and 5
# arrays in each group.

y<-matrix(rnorm(1000),ncol=10)

group<-factor(rep(c(1,2),each=5))
design<-model.matrix(~group)

# Fit linear model for differential variability
vfit<-varFit(y,design)

# Look at top table of results
topVar(vfit,coef=2)
```

---

varFit *Testing for differential variability*

Description

Fit linear model on mean absolute or squared deviations for each CpG given a series of methylation arrays

Usage

```
varFit(
  data,
  design = NULL,
  coef = NULL,
  type = NULL,
  trend = TRUE,
  robust = TRUE,
  weights = NULL
)

## S3 method for class 'MethylSet'
varFit(
  data,
  design = NULL,
  coef = NULL,
  type = NULL,
  trend = TRUE,
  robust = TRUE,
  weights = NULL
)

## S3 method for class 'DGEList'
```

```
varFit(
  data,
  design = NULL,
  coef = NULL,
  type = NULL,
  trend = TRUE,
  robust = TRUE,
  weights = NULL
)

## Default S3 method:
varFit(
  data,
  design = NULL,
  coef = NULL,
  type = NULL,
  trend = TRUE,
  robust = TRUE,
  weights = NULL
)
```

Arguments

data	Object of class MethylSet or matrix of M-values with rows corresponding to the features of interest such as CpG sites and columns corresponding to samples or arrays.
design	The design matrix of the experiment, with rows corresponding to arrays/samples and columns to coefficients to be estimated. Defaults to the unit vector.
coef	The columns of the design matrix containing the comparisons to test for differential variability. Defaults to all columns of design matrix.
type	Character string, "AD" for absolute residuals or "SQ" for squared residuals. Default is absolute.
trend	Logical, if true fits a mean variance trend on the absolute or squared deviations.
robust	Logical, if true performs robust empirical Bayes shrinkage of the variances for the moderated t statistics.
weights	Non-negative observation weights. Can be a numeric matrix of individual weights, of same size as the object matrix, or a numeric vector of array weights, or a numeric vector of gene/feature weights.

Details

This function depends on the limma package and is used to rank features such as CpG sites or genes in order of evidence of differential variability between different comparisons corresponding to the columns of the design matrix. A measure of variability is calculated for each CpG in each sample by subtracting out the group mean and taking the absolute or squared deviation. A linear model is then fitted to the absolute or squared deviations. The residuals of the linear model fit are subjected

to empirical Bayes shrinkage and moderated t statistics (Smyth, 2004) calculated. False discovery rates are calculated using the method of Benjamini and Hochberg (1995).

Please always specify the `coef` parameter in the call to `varFit`, which indicates which groups are to be tested for differential variability. If `coef` is not specified, then group means are estimated based on all the columns of the design matrix and subtracted out before testing for differential variability. If the design matrix contains nuisance parameters, then subsetting the design matrix columns by `coef` should remove these columns from the design matrix. If the design matrix includes an intercept term, this should be included in `coef`. The nuisance parameters are included in the linear model fit to the absolute or squared deviations, but should not be considered when subtracting group means to obtain the deviations. Note that design matrices without an intercept term are permitted, and specific contrasts tested using the function `contrasts.varFit`.

For methylation data, the analysis is performed on the M-values, defined as the log base 2 ratio of the methylated signal to the unmethylated signal. If a `MethylSet` object is supplied, M-values are extracted with an offset of 100 added to the numerator and denominator.

For testing differential variability on RNA-Seq data, a `DGEList` object can be supplied directly to the function. A `voom` transformation is applied before testing for differential variability. The weights calculated in `voom` are used in the linear model fit.

Since the output is of class `MArrayLM`, any functions that can be applied to fit objects from `lmFit` and `eBayes` can be applied, for example, `topTable` and `decideTests`.

Value

Produces an object of class `MArrayLM` (see `MArrayLM-class`) containing everything found in a fitted model object produced by `lmFit` and `eBayes` as well as a vector containing the sample CpG-wise variances and a matrix of LogVarRatios corresponding to the differential variability analysis.

`NULL`

`NULL`

`NULL`

Author(s)

Belinda Phipson

References

Phipson, B., and Oshlack, A. (2014). A method for detecting differential variability in methylation data shows CpG islands are highly variably methylated in cancers. *Genome Biology*, **15**:465.

Smyth, G.K. (2004). Linear models and empirical Bayes methods for assessing differential expression in microarray experiments. *Statistical Applications in Genetics and Molecular Biology*, Volume **3**, Article 3.

Smyth, G. K. (2005). Limma: linear models for microarray data. In: *Bioinformatics and Computational Biology Solutions using R and Bioconductor*. R. Gentleman, V. Carey, S. Dudoit, R. Irizarry, W. Huber (eds), Springer, New York, 2005.

Benjamini, Y., and Hochberg, Y. (1995). Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society Series*, B, **57**, 289-300.

See Also

contrasts.varFit, topVar, getLeveneResiduals, lmFit, eBayes, topTable, decideTests, voom

Examples

```
# Randomly generate data for a 2 group problem with 100 CpG sites and 5
# arrays in each # group.

y<-matrix(rnorm(1000),ncol=10)

group<-factor(rep(c(1,2),each=5))
design<-model.matrix(~group)

# Fit linear model for differential variability
vfit<-varFit(y,design,coef=c(1,2))

# Look at top table of results
topVar(vfit,coef=2)
```

Index