Package ‘missMethyl’

April 23, 2016

Type Package

Title Analysis of methylation array data

Version 1.4.0

Date 2015-09-22

Author Belinda Phipson and Jovana Maksimovic

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

Depends R (>= 2.3.0)

Imports limma, minfi, methylumi, IlluminaHumanMethylation450kmanifest, statmod, ruv, stringr, IlluminaHumanMethylation450kanno.ilmn12.hg19, org.Hs.eg.db

VignetteBuilder knitr

Suggests minfiData, BiocStyle, knitr, edgeR, tweeDEseqCountData

Description Normalisation and testing for differential variability and differential methylation for data from Illumina's Infinium HumanMethylation450 array. 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.

License GPL-2

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

NeedsCompilation no

R topics documented:

`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@mcri.edu.au>

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")

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

# 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, ce
```

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.

Author(s)

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

References

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

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")
}
```

`getINCs` *Extract intensity data for 613 Illumina negative controls found on 450k arrays.*

Description

Extracts the intensity data for the 613 Illumina negative controls found on 450k 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 450k 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 <jovana.maksimovic@mcri.edu.au>

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)
```

`gometh` *Gene ontology testing for 450K methylation data*

Description

Tests gene ontology enrichment for significant CpGs from Illumina's Infinium HumanMethylation450 array, taking into account the differing number of probes per gene present on the array.

Usage

```
gometh(sig.cpg, all.cpg = NULL, plot.bias = FALSE, prior.prob = TRUE)
```

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.
`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.

Details

This function takes a character vector of significant CpG sites, maps the CpG sites to Entrez Gene IDs, and tests for GO term enrichment using a hypergeometric test, taking into account the number of CpG sites per gene on the 450K array.

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 calls the `goana` function from the `limma` package (Ritchie et al. 2015). 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.

Genes associated with each CpG site are obtained from the annotation package `IlluminaHumanMethylation450kanno.ilmn`

`gometh` tests all GO terms, and false discovery rates are calculated using the method of Benjamini and Hochberg (1995).

Value

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

`Term`	GO term
`Ont`	ontology that the GO term belongs to. "BP" - biological process, "CC" - cellular component, "MF" - molecular function.
`N`	number of genes in the GO term

DE	number of genes that are differentially methylated
P.DE	p-value for over-representation of the GO term
FDR	False discovery rate

Author(s)

Belinda Phipson

References

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

goana

Examples

```
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, plot.bias=TRUE, prior.prob=TRUE)

# Total number of GO categories significant at 5% FDR
table(gst$FDR<0.05)

# Table of top GO results
topGO(gst)

# GO testing ignoring bias
gst.bias <- gometh(sig.cpg = sigcpgs, all.cpg = allcpgs, prior.prob=FALSE)
```

```
# Total number of GO categories significant at 5% FDR ignoring bias
table(gst.bias$FDR<0.05)

# Table of top GO results ignoring bias
topGO(gst.bias)
```

RUVadj *Adjust estimated variances*

Description

Calculate rescaled variances, empirical variances, etc. For use with RUV model fits produced using `RUVfit`.

Usage

```
RUVadj(fit, ebayes = TRUE, evar = FALSE, rsvar = FALSE, ...)
```

Arguments

`fit`	An object of class `MArrayLM` as returned by `RUVfit`.
`ebayes`	A logical variable. Should empirical bayes estimates be calculated?
`evar`	A logical variable. Should empirical variance estimates be calculated?
`rsvar`	A logical variable. Should rescaled variance estimates be calculated?
`...`	additional arguments that can be passed to `variance_adjust`. See linked function documentation for details.

Details

Adjust variance. By default only the empirical bayes method of Smyth (2004) is performed.

Value

An object of class `MArrayLM` containing:

`coefficients`	The estimated coefficients of the factor(s) of interest.
`sigma2`	Estimates of the features' variances.
`t`	t statistics for the factor(s) of interest.
`p`	P-values for the factor(s) of interest.
`multiplier`	The constant by which `sigma2` must be multiplied in order to get an estimate of the variance of `coefficients`
`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`.

The following items may or may not be present depending on the options selected when `RUVadj` was run:

`p.rsvar`	P-values, after applying the method of rescaled variances.
`p.evar`	P-values, after applying the method of empirical variances.
`p.ebayes`	P-values, after applying the empirical bayes method of Smyth (2004).
`p.rsvar.ebayes`	P-values, after applying the empirical bayes method of Smyth (2004) and the method of rescaled variances.
`p.BH`	P-values adjusted for false discovery rate (FDR) using the method of Benjamini and Hochberg (1995).
`p.rsvar.BH`	FDR-adjusted p-values, after applying the method of rescaled variances.
`p.evar.BH`	FDR-adjusted p-values, after applying the method of empirical variances.
`p.ebayes.BH`	FDR-adjusted p-values, after applying the empirical bayes method of Smyth (2004).
`p.rsvar.ebayes.BH`	FDR-adjusted p-values, after applying the empirical bayes method of Smyth (2004) and the method of rescaled variances.

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(data=Mval, design=design, coef=2, ctl=ctl)
fit2 = RUVadj(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(data, design, coef, ctl, method=c("inv", "rinv", "ruv4", "ruv2"),
k = NULL, ...)
```

Arguments

`data`	numeric `matrix` 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.
`coef`	integer, column of the design matrix containing the comparison to test for differential methylation. Default is the last colum of the design matrix.
`ctl`	logical vector, `length == nrow(data)`. Features that are to be used as negative control variables are indicated as TRUE, all other features are FALSE.
`method`	character string, indicates which RUV method should be used. Default method is `RUVinv`.
`k`	integer, required if `method` is "ruv2" or "ruv4". Indicates the number of unwanted factors to use. Can be 0.
`...`	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` and `limma` packages 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

An object of class `MArrayLM` (see `MArrayLM-class`) containing:

`coefficients`	The estimated coefficients of the factor(s) of interest.
`sigma2`	Estimates of the features' variances.
`t`	t statistics for the factor(s) of interest.
`p`	P-values for the factor(s) of interest.
`multiplier`	The constant by which `sigma2` must be multiplied in order to get an estimate of the variance of `coefficients`
`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.

Author(s)

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

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(data=Mval, design=design, coef=2, ctl=ctl)
fit2 = RUVadj(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)
```

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`

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 `<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 `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))
}
```

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`.

Usage

```
topRUV(fit, number=10, p.value.cut = 1,
cut.on = c("p.ebayes.BH","p.BH","p.rsvar.BH","p.evar.BH","p.rsvar.ebayes.BH"),
sort.by = c("p.ebayes.BH","p.BH","p.rsvar.BH","p.evar.BH","p.rsvar.ebayes.BH"))
```

Arguments

`fit`	An object containing a linear model fit produced by `RUVfit`, followed by `RUVadj`. The fit object should be of class `MArrayLM`.
`number`	integer, maximum number of genes to list. Default is 10.
`p.value.cut`	numeric, cutoff value for adjusted p-values. Only features with lower p-values are listed. Must ve between 0 and 1. Default is 1.
`cut.on`	numeric, the type of adjusted p-value that the cutoff should be applied to. Default is `p.ebayes.BH`. Other options are: `p.BH`, `p.rsvar.BH`, `p.evar.BH` or `p.rsvar.ebayes.BH`.

`sort.by`	character string, the type of adjusted p-value that should be used for sorting. Default is `p.ebayes.BH`. Other options are: `p.BH`, `p.rsvar.BH`, `p.evar.BH` or `p.rsvar.ebayes.BH`.

Details

This function summarises the results of a differential methylation analysis performed using `RUVfit`, followed by `RUVadj`. The top ranked CpGs are selected by first ranking the adjusted p-values (Default: `p.ebayes.BH`), then ranking the raw p-values (Default: `p.ebayes`).

Value

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

`coefficients`	The estimated coefficients of the factor(s) of interest.
`sigma2`	Estimates of the features' variances.
`t`	t statistics for the factor(s) of interest.
`p`	P-values for the factor(s) of interest.
`multiplier`	The constant by which `sigma2` must be multiplied in order to get an estimate of the variance of `coefficients`
`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`.

The following columns may or may not be present depending on the options selected when `RUVadj` was run:

`p.rsvar`	P-values, after applying the method of rescaled variances.
`p.evar`	P-values, after applying the method of empirical variances.
`p.ebayes`	P-values, after applying the empirical bayes method of Smyth (2004).
`p.rsvar.ebayes`	P-values, after applying the empirical bayes method of Smyth (2004) and the method of rescaled variances.
`p.BH`	P-values adjusted for false discovery rate (FDR) using the method of Benjamini and Hochberg (1995).
`p.rsvar.BH`	FDR-adjusted p-values, after applying the method of rescaled variances.

`p.evar.BH`	FDR-adjusted p-values, after applying the method of empirical variances.
`p.ebayes.BH`	FDR-adjusted p-values, after applying the empirical bayes method of Smyth (2004).
`p.rsvar.ebayes.BH`	FDR-adjusted p-values, after applying the empirical bayes method of Smyth (2004) and the method of rescaled variances.

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(data=Mval, design=design, coef=2, ctl=ctl)
fit2 = RUVadj(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)
```

Arguments

`data`	object of class `MethylSet` or `matrix` 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.
`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).

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.

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