Package 'lmdme'

October 9, 2015

Type Package

Title Linear Model decomposition for Designed Multivariate Experiments

Version 1.10.0

Date 2014-01-31

Author Cristobal Fresno and Elmer A. Fernandez

Maintainer Cristobal Fresno <cfresno@bdmg.com.ar>

Description linear ANOVA decomposition of Multivariate Designed Experiments implementation based on limma lmFit. Features: i)Flexible formula type interface, ii) Fast limma based implementation, iii) p-values for each estimated coefficient levels in each factor, iv) F values for factor effects and v) plotting functions for PCA and PLS.

License GPL (>=2)

URL http://www.bdmg.com.ar/?page_id=38

Imports stats, methods, limma

Depends R (>= 2.14.1), pls, stemHypoxia

Enhances parallel

Collate 'lmdme-Class.R' 'lmdme-lmdme.R' 'lmdme-getters.R' 'lmdme-printshow.R' 'lmdme-padjust.R' 'lmdme-leverage.R' 'lmdme-permutation.R' 'lmdme-decomposition.R' 'lmdme-biplot.R' 'lmdme-screeplot.R' 'lmdme-loadingplot.R'

biocViews Microarray, OneChannel, TwoChannel, Visualization, DifferentialExpression, ExperimentData, Cancer

NeedsCompilation no

R topics documented:

`biplot` *Plot a* `biplot` *of a lmdme object*

Description

Plot a biplot over each decomposed "pca" or "plsr" present in lmdme component object's slot.

Usage

```
  ## S4 method for signature 'lmdme'
biplot(x, comp=1:2, xlab=NULL,
  ylab=NULL, term=NULL, mfcol, xlabs, ylabs, which, ...)
```

Arguments

`x`	lmdme class object.
`comp`	a two component vector with the PC components to plot. Default comp=1:2.
`xlab`	character for the x-label title for PCA biplots.
`ylab`	character for the y-label title for PCA biplots.
`term`	character with the corresponding term/s for biploting. Default value is NULL in order to obtain every available biplot/s.
`mfcol`	numeric vector for par layout. If missing mfcol=c(1,2) will be used if more than one biplot is available. Use mfcol==NULL to override par call inside biplot function.
`xlabs,ylabs`	vector of character strings to label the first/second set of points. The default is to use dimname of "x"/"y", or "1:n" if the dimname is NULL for the respective set of points. If a single character is passed e.g. "o", the same character is used for all the points.
`which`	character to indicate the type of biplot to use when plsr decomposition is applied. Default value is "x" (X scores and loadings), "y" for (Y scores and loadings), "scores" (X and Y scores) or "loadings" (X and Y loadings). See `biplot.mvr` for details.
`...`	additional parameters for `biplot.prcomp`(pca) or `biplot.mvr`(plsr)

Value

plotted biplot/s of the component/s of the given lmdme object. If par() is called before this function, the biplots can be arranged in the same window

Author(s)

Cristobal Fresno and Elmer A Fernandez

See Also

prcomp, plsr, biplot.princomp, biplot.mvr

Examples

```
{
data(stemHypoxia)

##Just to make a balanced dataset in the Fisher sense (2 samples per
## time*oxygen levels)
design<-design[design$time %in% c(0.5,1,5) & design$oxygen %in% c(1,5,21), ]
design$time<-as.factor(design$time)
design$oxygen<-as.factor(design$oxygen)
rownames(M)<-M[, 1]

#Keeping appropriate samples only
M<-M[, colnames(M) %in% design$samplename]

##ANOVA decomposition
fit<-lmdme(model=~time+oxygen+time:oxygen, data=M, design=design)

##ASCA for all the available terms, over those subjects/genes where at least
##one interaction coefficient is statistically different from zero (F-test
##on coefficients).
id<-F.p.values(fit, term="time:oxygen")<0.001
decomposition(fit, decomposition="pca",scale="row",subset=id)

## Not run:
##Does not call par inside
par(mfrow=c(2,2))
biplot(fit, xlabs="o", mfcol=NULL)

##Just the term of interest
biplot(fit, xlabs="o", term="time")

##In separate graphics
biplot(fit, xlabs="o", term=c("time", "oxygen"), mfcol=c(1,1))

##All terms in the same graphic
biplot(fit, xlabs="o", mfcol=c(1,3))

## End(Not run)
}
```

```
##Now using plsr on interaction coefficients
decomposition(fit, decomposition="plsr", term="time:oxygen", scale="row",
subset=id)

## Not run:
par(mfrow=c(2,2))

##plsr biplot by default which="x"
biplot(fit, which="x", mfcol=NULL)

##Other alternatives to which
biplot(fit, which="y", mfcol=NULL)
biplot(fit, which="scores", mfcol=NULL)
biplot(fit, which="loadings", mfcol=NULL, xlabs="o")

## End(Not run)
```

`decomposition` `decomposition` *of lmdme object*

Description

This function calculates the decomposition of variance or covariance structure using Principal Component Analysis (PCA) or Partial Least Squared Regression (PLSR), on the ANOVA decomposed lmdme object. In this context, in a two factor experimental design with interaction, the linear model of the i-th observation (gene) can be written:
$X = \mu + X_A + X_B + X_{AB} + \epsilon$
where

- X stands for the observed value.
- the intercept μ.
- X_A, X_B and X_{AB} are the first, second and interaction coefficients respectively.
- The error term ϵ $N(0, \sigma^2)$.

The model is iteratively decomposed in a step by step fashion, decomposing one term each time by calling `lmdme` constructor:

1. Step 1: $X = \mu + E_1$
2. Step 2: $E_1 = X_A + E_2$
3. Step 3: $E_2 = X_B + E_3$
4. Step 4: $E_3 = X_{AB} + E_4$

Then, if we apply PCA on the i-th step using E_{i-1} matrix it is known as *APCA* but if applied on the coefficients X_i it is called *ASCA*. The same decomposition schema can also be used with PLSR.

Arguments

`object`	lmdme class object.
`decomposition`	character to indicate the decomposition to be carried out, i.e., *"pca"* or *"plsr"*. Default value is "pca".
`term`	character specifying the model term to perform the decomposition (e.g. "time" or "time:concentration" for interaction term). If the term is not specified (i.e. missing) it performs the analysis over all the model terms.
`subset`	subset of individuals (rows) to be included in the analysis. By default all the individuals are included.
`type`	character to indicate on which regression matrix ("coefficient" or "residual") the decomposition will be performed. The intercept term is not included in the results, as it can be directly analyzed with the original M data.frame. Default value is "coefficient" a.k.a. ASCA. Note that "residual" performs PCA or PLS on the i-th residual $E_{i-1} = X_i + E_i$ and not the residuals of the i-th model E_i.
`scale`	character "row", "column" or "none" to indicate if the matrix should be scaled by the row, column or not respectively. Default value is "none".
`Omatrix`	the output matrix for PLSR only. If the parameter is missing, the output matrix will be an identity matrix for the ASCA. Otherwise, is the design matrix corresponding to the specified term for APCA.
`...`	additional parameters for `prcomp` or `plsr` functions, according to the decomposition call.

Value

Internal update of the "components" slot of the lmdme object, which is a list of `prcomp` or a list of mvr (`plsr`) objects using the given term parameter. If missing(term), the length of the list equals the number of decomposed models minus the Intercept term for coefficients or the length of decomposed models for residual decomposition.

Author(s)

Cristobal Fresno and Elmer A Fernandez

References

1. Smilde AK, Jansen JJ, Hoefsloot HCJ, Lamer RAN, Van der Greef J, Timmerman ME (2005) ANOVA-simultaneous component analysis (ASCA): a new tool for analyzing designed metabolomics data, Bioinformatics 21,13,3043 DOI:/10.1093/bioinformatics/bti476
2. Zwanenburg G, Hoefsloot HCJ, Westerhuis JA, Jansen JJ, Smilde AK (2011) ANOVA Principal component analysis and ANOVA-simultaneous component analysis: a comparison J. Chemometrics 25:561-567 DOI:10.1002/cem.1400
3. Cristobal Fresno, Monica G. Balzarini, Elmer A. Fernandez (2014) lmdme: Linear Models on Designed Multivariate Experiments in R, Journal of Statistical Software, 56(7), 1-16, http://www.jstatsoft.org/v56/i07/.

See Also

prcomp, plsr

Examples

```
{
data(stemHypoxia)

##Just to make a balanced dataset in the Fisher sense (2 samples per
## time*oxygen levels)
design<-design[design$time %in% c(0.5,1,5) & design$oxygen %in% c(1,5,21), ]
design$time<-as.factor(design$time)
design$oxygen<-as.factor(design$oxygen)
rownames(M)<-M[, 1]

#Keeping appropriate samples only
M<-M[, colnames(M) %in% design$samplename]

##ANOVA decomposition
fit<-lmdme(model=~time+oxygen+time:oxygen, data=M, design=design)

##Just a copy of the same fit object and to perform analysis on those
##subjects/genes where at least one interaction coefficient is statistically
##different from zero (F-test on the coefficients).
asca<-fit
apca<-fit
id<-F.p.values(fit, term="time:oxygen")<0.001

##ASCA and APCA decomposition for every available term.
decomposition(asca, decomposition="pca", subset=id, scale="row")
decomposition(apca, decomposition="pca", subset=id, scale="row",
type="residual")

##Let's get the components for asca/apca decomposed objects
asca<-components(asca)
apca<-components(apca)

##Now let's try the PLSR decomposition for residuals and coefficients
plsr.residuals<-fit
plsr.coefficients<-fit
decomposition(plsr.coefficients, decomposition="plsr", subset=id,
scale="row")
decomposition(plsr.residuals, decomposition="plsr", subset=id, scale="row",
type="residual")

##Obtain the coefficients for decomposed plsr objects
##(coefficients/residuals)
plsr.coefficients<-components(plsr.coefficients)
plsr.residuals <- components(plsr.residuals)
}
```

fitted.values *Getters for lmdme object*

Description

To obtain lmdme slot information, according to the given function call (see Values). If a term parameter is not specified, it will return all the available terms. Otherwise, just the one specified.

Usage

```
  ## S4 method for signature 'lmdme'
fitted.values(object, term=NULL,
  drop=TRUE)

  ## S4 method for signature 'lmdme'
fitted(object, term=NULL, drop=TRUE)

  ## S4 method for signature 'lmdme'
coef(object, term=NULL, drop=TRUE)

  ## S4 method for signature 'lmdme'
coefficients(object, term=NULL,
  drop=TRUE)

  ## S4 method for signature 'lmdme'
resid(object, term=NULL, drop=TRUE)

  ## S4 method for signature 'lmdme'
residuals(object, term=NULL, drop=TRUE)

  F.p.values(object, term=NULL, drop=TRUE)

  ## S4 method for signature 'lmdme'
F.p.values(object, term=NULL,
  drop=TRUE)

  p.values(object, term=NULL, drop=TRUE)

  ## S4 method for signature 'lmdme'
p.values(object, term=NULL, drop=TRUE)

  modelDecomposition(object, term=NULL, drop=TRUE)

  ## S4 method for signature 'lmdme'
modelDecomposition(object, term=NULL,
  drop=TRUE)
```

```
  components(object, term=NULL, drop=TRUE)

  ## S4 method for signature 'lmdme'
components(object, term=NULL,
  drop=TRUE)

  componentsType(object)

  ## S4 method for signature 'lmdme'
componentsType(object)

  model(object)

  ## S4 method for signature 'lmdme'
model(object)

  design(object)

  ## S4 method for signature 'lmdme'
design(object)
```

Arguments

`object`	lmdme class object.
`term`	character with the corresponding term/s to return. Default value is NULL in order to return every available term/s.
`drop`	should try to drop list structure if length==1? Default value is TRUE

Value

according to the call one of the following objects can be returned

`design`	experiment design data.frame used.
`model`	decomposed formula used.
`modelDecomposition`	list of decomposed model formulas.
`residuals, resid, coef, coefficients, fitted, fitted.values, p.values or F.p.values`	list of appropriate slot where each item is a matrix that will have G rows (individuals) x k columns (levels of the corresponding model term).
`components`	list with corresponding PCA or PLSR terms according to the decomposition function call.
`componentsType`	character name vector with the information of the component calculations.

Author(s)

Cristobal Fresno and Elmer A Fernandez

See Also

lmdme, decomposition, print, show

Examples

```
{
data(stemHypoxia)

##Just to make a balanced dataset in the Fisher sense (2 samples per
## time*oxygen levels)
design<-design[design$time %in% c(0.5, 1, 5) & design$oxygen %in% c(1,5,21),]
design$time<-as.factor(design$time)
design$oxygen<-as.factor(design$oxygen)
rownames(M)<-M[, 1]

##Keeping appropriate samples only
M<-M[, colnames(M) %in% design$samplename]

##ANOVA decomposition
fit<-lmdme(model=~time+oxygen+time:oxygen, data=M, design=design)

##Let's inspect how the decomposition process was carried out:
##a) The model formula used
##b) The design data.frame used
##c) The decomposition itself
fit.model<-model(fit)
fit.design<-design(fit)
fit.modelDecomposition<-modelDecomposition(fit)

##Getting the specific "time" term coefficients, p-values or F-values.
## Omit "term" parameter for all available terms.
timeCoef<-coef(fit,term="time")
fit.p.values<-p.values(fit,term="time")
fit.f.values<-F.p.values(fit,term="time")

##Getting the residuals or fitted values, for the interaction "time:oxygen"
## term. Omit "term" parameter for all available terms.
interactionResid<-resid(fit, term="time:oxygen")
interactionFit<-fitted(fit, term="time:oxygen")
}
```

`leverage` `leverage` *test of lmdme objects*

Description

This function calculates the leverage test for each individual using the Principal Component Analysis (comps function) on the coefficients of the given decomposed model term.

Arguments

object	lmdme class object.
comps	a numeric vector indicating the PCA component indexes to keep. Default the first two components (1:2).
term	a character specifying the model term.
level	the quantile level. Default value 0.95

Value

data.frame with the following fields

leverage	numeric for the corresponding row leverage
over	logical indicating if the leverage > quantile(leverage,level) for the given decomposed term

Author(s)

Cristobal Fresno and Elmer A Fernandez

References

Tarazona S, Prado-Lopez S, Dopazo J, Ferrer A, Conesa A, Variable Selection for Multifactorial Genomic Data, Chemometrics and Intelligent Laboratory Systems, 110:113-122 (2012)

See Also

prcomp, quantile

Examples

```
{
data(stemHypoxia)

##Just to make a balanced dataset in the Fisher sense (2 samples per
## time*oxygen levels)
design<-design[design$time %in% c(0.5, 1, 5) & design$oxygen %in% c(1,5,21),]
design$time<-as.factor(design$time)
design$oxygen<-as.factor(design$oxygen)
rownames(M)<-M[, 1]

##Keeping appropriate samples only
M<-M[, colnames(M) %in% design$samplename]

##ANOVA decomposition
fit<-lmdme(model=~time+oxygen+time:oxygen, data=M, design=design)

##Leverages for the first two Principal Components and q95 (default value).
##Leverages for the first three Principal Components and q99.
leverages2PCDefault<-leverage(fit, term="time:oxygen")
leverages3PCq99<-leverage(fit, comps=1:3, term="time:oxygen", level=0.99)
}
```

`lmdme` *High level constructor of lmdme class object*

Description

Linear model ANOVA decomposition of Designed Multivariate Experiments based on limma `lmFit` implementation. For example in a two factor experimental design with interaction, the linear model of the i-th observation (gene) can be written:
$X = \mu + A + B + AB + \epsilon$
where

- X stands for the observed value
- the intercept μ
- A, B and AB are the first, second and interaction terms respectively
- The error term ϵ $N(0, \sigma^2)$.

The model is iteratively decomposed in a step by step fashion decomposing one term each time:

1. The intercept is estimated using $X = \mu + E_1$
2. The first factor (A) using $E_1 = A + E_2$
3. The second factor (B) using $E_2 = B + E_3$
4. The interaction (AB) using $E_3 = AB + E_4$.

For each decomposed step the model, residuals, coefficients, p-values and F-value are stored in a list container, so their corresponding length is equal to the number of model terms + 1 (the intercept).

Arguments

`model`	formula object to carry out the decomposition.
`data`	matrix or data.frame with individuals/genes (per rows) and samples/conditions (per columns).
`design`	data.frame with the design of the experiment, (rows) samples/conditions as in data columns and as many columns to indicate the factors present in each sample.
`Bayes`	Should limma estimate empirical Bayes statistics, i. e., moderated t-statistics? Default value is FALSE.
`verbose`	Should the process progress be printed? Default value is FALSE.
`...`	Additional parameters for `lmFit` function.

Value

`lmdme`	lmdme class object with the corresponding completed slots according to the given model

Note

use lmdme high level constructor for the creation of the class instead of directly calling its constructor by means of new.

Author(s)

Cristobal Fresno and Elmer A Fernandez

References

1. 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, pages 397–420.
2. Cristobal Fresno, Monica G. Balzarini, Elmer A. Fernandez (2014) lmdme: Linear Models on Designed Multivariate Experiments in R, Journal of Statistical Software, 56(7), 1-16, http://www.jstatsoft.org/v56/i07/.

See Also

decomposition, lmFit

Examples

```
{
data(stemHypoxia)

##Just to make a balanced dataset in the Fisher sense (2 samples per
## time*oxygen levels)
design<-design[design$time %in% c(0.5,1,5) & design$oxygen %in% c(1,5,21), ]
design$time<-as.factor(design$time)
design$oxygen<-as.factor(design$oxygen)
rownames(M)<-M[, 1]

#Keeping appropriate samples only
M<-M[, colnames(M) %in% design$samplename]

##ANOVA decomposition
fit<-lmdme(model=~time+oxygen+time:oxygen, data=M, design=design)
}
```

`lmdme-class` *lmdme S4 class: Linear Model decomposition for Designed Multivariate Experiments.*

Description

Linear Model ANOVA decomposition of Designed Multivariate Experiments based on limma lmFit implementation. For example in a two factor experimental design with interaction, the linear model of the i-th observation (gene) can be written:
$X = \mu + A + B + AB + \epsilon$
where

- X stands for the observed value
- The intercept μ
- A, B and AB are the first, second and interaction terms respectively
- The error term ϵ $N(0, \sigma^2)$.

The model is iterative decomposed in a step by step fashion decomposing one term at each time:

1. The intercept is estimated using $X = \mu + E_1$
2. The first factor (A) using $E_1 = A + E_2$
3. The second factor (B) using $E_2 = B + E_3$
4. The interaction (AB) using $E_3 = AB + E_4$.

For each decomposed step the model, residuals, coefficients, p-values and F-values are stored in a list container, so their corresponding length is equal to the number of model terms + 1 (the intercept).

Features

1. Flexible formula type interface,
2. Fast limma based implementation based on lmFit,
3. p values for each estimated coefficient levels in each factor
4. F values for factor effects
5. Plotting functions for PCA and PLS.

Slots

- design: data.frame with experimental design.
- model: formula with the designed model to be decomposed.
- modelDecomposition: list with the model formula obtained for each decomposition step.
- residuals: list of residual matrices G rows(genes) x N columns (arrays-designed measurements).
- coefficients: list of coefficient matrices. Each matrix will have G rows(genes) x k columns(levels of the corresponding model term).
- p.values: list of p-value matrices.
- F.p.values: list with corresponding F-p-values vectors for each individual.
- components: list with corresponding PCA or PLS components for the selected term/s.
- componentsType: name character vector to keep process trace of the variance/covariance components slot: decomposition ("pca" or "pls"), type ("apca" for ANOVA-PCA or "asca" for ANOVA-SCA) and scale ("none", "row" or "column")

lmdme-general-functions

print, show Basic output for lmdme class

summary Basic statistics for lmdme class

design, model, modelDecomposition, residuals and coefficients Getters for their respective slots.

p.values, F.p.values, components and componentsType Getters for their respective slots.

ANOVA-linear-decomposition-functions

lmdme Function that produces the complete ANOVA decomposition based on model specification through a formula interface. Technically it's a high level wrapper of the initialize function.

modelDecomposition Getter for the used decomposed formula in each step

p.adjust Adjust coefficients p-values for the Multiple Comparison Tests.

Fpvalues, pvalues Getters for the corresponding associated decomposed model coefficient statistics in each step, for each observation.

residuals, resid, coef, coefficients, fitted.values, fitted Getters for the corresponding decomposed model in each step.

permutation Produces the specified lmdme in addition to the required permuted objects (sampling the columns of data), using the same parameters to fit the model.

variance-covariance-decomposition-functions

decomposition Function to perform PCA or PLS on the ANOVA decomposed terms. PCA can be performed on E_1, E_2 or E_3 and it is referred to, as ANOVA-PCA (APCA) but, if it is performed on the coefficients it is referred to, as ANOVA-SCA (ASCA). On the other hand PLSR is based on pls library and if it is performed on coefficients (ASCA like) it uses the identity matrix for output co-variance maximization or can be carried out on the $E_{1,2or3}$ (APCA like) using the design matrix as the output.

components Getter for PCA or PLS decomposed models.

componentsType Getter for componentsType slot.

leverage Leverage calculation on PCA (APCA or ASCA) terms.

biplot Biplots for PCA or PLSR decomposed terms.

screeplot Screeplot on each PCA decomposed term.

loadingplot Loadingplot for PCA interaction terms.

Author(s)

Cristobal Fresno and Elmer A Fernandez

References

1. Smilde AK, Jansen JJ, Hoefsloot HCJ, Lamer RAN, Van der Greef J, Timmerman ME (2005) ANOVA-simultaneaus component analysis (ASCA): a new tool for analyzing designed metabolomics data, Bioinformatics 21,13,3043 DOI:/10.1093/bioinformatics/bti476

2. Zwanenburg G, Hoefsloot HCJ, Westerhuis JA, Jansen JJ, Smilde AK (2011) ANOVA.Principal component analysis and ANOVA-simultaneaus component analysis: a comparison J. Chemometrics 25:561-567 DOI:10.1002/cem.1400
3. Tarazona S, Prado-Lopez S, Dopazo J, Ferrer A, Conesa A (2012) Variable Selection for Multifactorial Genomic Data, Chemometrics and Intelligent Laboratory Systems, 110:113-122

See Also

lmdme, decomposition, biplot, loadingplot and additional related lmdme class functions.

`loadingplot` `loadingplot` *of interaction PCA decomposed lmdme object*

Description

This function plots the PCA loadings for a given interaction (A:B) lmdme object's components slot, for the given "pc" component. The user can choose which term (A or B) is used for x-axis and y-axis functions (B or A) respectively.

Usage

```
  ## S4 method for signature 'lmdme'
loadingplot(object, term.x, term.y,
  pc=1, ord.x, col, ...)
```

Arguments

`object`	lmdme class object.
`term.x,term.y`	character indicating the model principal factor for the interaction term (term.x:term.y or term.y:term.x) for the corresponding x or y axis.
`pc`	integer indicating which principal component loading is to be plotted on the y-axis. Default value is 1.
`col`	which color to use for each level present in term.y.
`ord.x`	numeric indicating the term.x levels order, for plotting purposes. If missing, the levels order is used.
`...`	additional parameters for matplot.

Value

loading plot of the selected interaction (term.x:term.y) lmdme object's components slot, if PCA decomposition was applied.

Author(s)

Cristobal Fresno and Elmer A Fernandez

Examples

```
{
data(stemHypoxia)

##Just to make a balanced dataset in the Fisher sense (2 samples per
## time*oxygen levels)
design<-design[design$time %in% c(0.5,1,5) & design$oxygen %in% c(1,5,21), ]
design$time<-as.factor(design$time)
design$oxygen<-as.factor(design$oxygen)
rownames(M)<-M[, 1]

#Keeping appropriate samples only
M<-M[, colnames(M) %in% design$samplename]

##ANOVA decomposition
fit<-lmdme(model=~time+oxygen+time:oxygen, data=M, design=design)

##ASCA for all the available terms, on those subjects/genes where at least
##one interaction coefficient is statistically different from zero (F-test
##on the coefficients).
id<-F.p.values(fit, term="time:oxygen")<0.001
decomposition(fit, decomposition="pca", scale="row", subset=id)

## Not run:
loadingplot(fit, term.x="time", term.y="oxygen")

##Or change the axis order
loadingplot(fit, term.x="oxygen", term.y="time")

##Or change the PC to display
loadingplot(fit, term.x="time", term.y="oxygen", pc=2)

##Or the order of x-levels
loadingplot(fit, term.x="time", term.y="oxygen", ord.x=3:1)

## End(Not run)
}
```

`p.adjust` `p.adjust` *of p-values for Multiple Test Comparison Corrections*

Description

Given a set of p-values, returns adjusted p-values using one of several methods.

Usage

```
  ## S4 method for signature 'lmdme'
p.adjust(p, term=NULL,
  method=p.adjust.methods, drop=TRUE)
```

Arguments

p	numeric vector of p-values as in stats::p.adjust or lmdme class object.
method	correction method available in p.adjust.methods.
term	character with the corresponding term to return.
...	other arguments.
drop	should try to drop the list structure if length==1? Default value is TRUE

Value

according to the call, one of the following objects can be returned

numeric	vector of adjusted p-values.
matrix	for lmdme object If term!=NULL, the corresponding character is looked up within the list of p.values returning the associated matrix of G rows (individuals) x k columns (levels of the corresponding model term) with the adjusted p-values.

Author(s)

Cristobal Fresno and Elmer A Fernandez

See Also

p.adjust, p.adjust.methods

Examples

```
{
data(stemHypoxia)

##Just to make a balanced dataset in the Fisher sense (2 samples per
## time*oxygen levels)
design<-design[design$time %in% c(0.5, 1, 5) & design$oxygen %in% c(1,5,21),]
design$time<-as.factor(design$time)
design$oxygen<-as.factor(design$oxygen)
rownames(M)<-M[, 1]

##Keeping appropriate samples only
M<-M[, colnames(M) %in% design$samplename]

##ANOVA decomposition
fit<-lmdme(model=~time+oxygen+time:oxygen, data=M, design=design)

##Adjust p-values only on the interaction p.values using false discovery rate
## method
pInteraction<-p.values(fit, term="time:oxygen")
FDRInteraction<-p.adjust(fit, term="time:oxygen", method="fdr")
corrected<-sum(pInteraction < 0.05) - sum(FDRInteraction < 0.05)
}
```

permutation permutation *of the specified lmdme object*

Description

Produces the specified lmdme plus the required permuted objects (sampling the columns), using the same parameters to fit the additional models.

Usage

```
  ## S4 method for signature 'formula,data.frame,data.frame'
permutation(model,data,design,Bayes=FALSE,verbose=FALSE,NPermutations=100,nCpus=1,...)
```

Arguments

model	formula object to carry out the decomposition.
data	data.frame with individuals (rows) and samples/conditions (columns)
design	data.frame with the design of the experiment, (rows) samples/conditions as in data columns and as many columns to indicate the factors present in each sample.
Bayes	Should limma estimate empirical Bayes statistics, i.e., moderated t-statistics? Default value is FALSE.
verbose	Should the process progress be printed? Default value is FALSE.
NPermutations	number of permutations to be calculated. Default value is 100.
nCpus	number of cores to be used. Default value is 1, i.e. sequential calculation.
...	Additional parameters for the lmFit function.

Value

list	contains the original lmdme object plus the required amount of permuted versions.

Author(s)

Cristobal Fresno and Elmer A Fernandez

See Also

lmdme

Examples

```
{
data(stemHypoxia)

##Just to make a balanced dataset in the Fisher sense (2 samples per
## time*oxygen levels)
design<-design[design$time %in% c(0.5, 1, 5) & design$oxygen %in% c(1,5,21),]
design$time<-as.factor(design$time)
design$oxygen<-as.factor(design$oxygen)
rownames(M)<-M[, 1]

##Keeping appropriate samples only
M<-M[, colnames(M) %in% design$samplename]

##Just to test if it works. In a real scenario, use NPermutations >= 100 if
##the conditions (columns) of M allow it. Verbose parameter is FALSE by
##default
permuted<-permutation(model=~time*oxygen, data=M, design=design,
NPermutations=2, nCpus=3)
}
```

---

print Show, Print *or* Summary *a lmdme object*

Description

Generic Show/Print/Summary method for lmdme class output visualization.

Usage

```
  ## S4 method for signature 'lmdme'
print(x, term)

  ## S4 method for signature 'lmdme'
show(object)

  ## S4 method for signature 'lmdme'
summary(object)
```

Arguments

x	lmdme class object.
object	lmdme class object.
term	character with the corresponding term to return. Default value is NULL to return every decomposed term (if more than one is available).

Value

according to the call

`print, show or summary:`
: console output text with increasing detail of lmdme object.

`show or summary`
: console output text of the lmdme object, plus a data.frame with model decomposition summary data.

Author(s)

Cristobal Fresno and Elmer A Fernandez

See Also

`lmdme`, `coef`, `resid`, `fitted`, `modelDecomposition`, `components`, `componentsType`

Examples

```
{
data(stemHypoxia)

##Just to make a balanced dataset in the Fisher sense (2 samples per
## time*oxygen levels)
design<-design[design$time %in% c(0.5, 1, 5) & design$oxygen %in% c(1,5,21),]
design$time<-as.factor(design$time)
design$oxygen<-as.factor(design$oxygen)
rownames(M)<-M[, 1]

##Keeping appropriate samples only
M<-M[, colnames(M) %in% design$samplename]

##ANOVA decomposition
fit<-lmdme(model=~time+oxygen+time:oxygen, data=M, design=design)
}
## Not run:
#equivalent to call show(fit)
fit
print(fit)
summary(fit)

## End(Not run)
```

`screeplot` *Plot a* `screeplot` *of a PCA decomposed lmdme object*

Description

Screeplot on each decomposed "pca" model present in lmdme components slot.

Usage

```
  ## S4 method for signature 'lmdme'
screeplot(x, independent=TRUE,
  col=seq(along=components(x)), npcs, term=NULL, mfcol,
  ...)
```

Arguments

x	lmdme class object.
independent	logical indicating whether the screeplots should be plotted together. Default value is FALSE.
col	which color to use for each decomposed model. Default value seq(along= components(x)).
npcs	integer with the number of components to plot. By default all present components are plotted.
term	character with the corresponding term/s for biploting. Default value is NULL in order to obtain every available biplot/s.
mfcol	numeric vector for par layout. If missing, mfcol=c(1,2) will be used if more than one biplot is available. Use mfcol==NULL to override par call inside biplot function.
...	additional parameters for screeplot or plot/lines according to independent FALSE or TRUE respectively.

Value

plotted screeplot/s of the components slot if PCA decomposition was applied.

Author(s)

Cristobal Fresno and Elmer A Fernandez

See Also

stats::screeplot

Examples

```
{
data(stemHypoxia)

##Just to make a balanced dataset in the Fisher sense (2 samples per
## time*oxygen levels)
design<-design[design$time %in% c(0.5,1,5) & design$oxygen %in% c(1,5,21), ]
design$time <-as.factor(design$time)
design$oxygen<-as.factor(design$oxygen)
rownames(M)<-M[, 1]

#Keeping appropriate samples only
```

```
M<-M[, colnames(M) %in% design$samplename]

##ANOVA decomposition
fit<-lmdme(model=~time+oxygen+time:oxygen, data=M, design=design)

##ASCA for all the available terms, on those subjects/genes where at least
##one interaction coefficient is statistically different from zero (F-test
##on the coefficients).
id<-F.p.values(fit,term="time:oxygen")<0.001
decomposition(fit, decomposition="pca", scale="row", subset=id)

## Not run:
par(mfrow=c(2,2))

##Does not call par inside
screeplot(fit,mfcol=NULL)

##Just the term of interest
screeplot(fit,term="time")

##In separate graphics
screeplot(fit,term=c("time","oxygen"),mfcol=c(1,1))

##All term in the same graphic device
screeplot(fit,mfcol=c(1,3))

##All in the same graphic
screeplot(fit,independent=FALSE)

## End(Not run)
}
```

Index