Package 'phia'

May 30, 2025

Version 0.3-2

Date 2025-05-30

Title Post-Hoc Interaction Analysis

Description Analysis of terms in linear, generalized and mixed linear models, on the basis of multiple comparisons of factor contrasts. Specially suited for the analysis of interaction terms.

Depends car, graphics, stats

Suggests nlme, lme4

Imports Matrix, grDevices, methods, utils

License GPL (>= 3)

URL https://github.com/heliosdrm/phia

LazyData yes

NeedsCompilation no

Author Helios De Rosario-Martinez [aut, cre],
John Fox [ctb],
R Core Team [ctb],
Chalmers Phil [ctb]

Maintainer Helios De Rosario-Martinez <helios.derosario@gmail.com>

Repository CRAN

Date/Publication 2025-05-30 20:10:02 UTC

Contents

`phia-package` *Post-Hoc Interaction Analysis*

Description

Analysis of the expected values and other terms of in linear, generalized, and mixed linear models, on the basis of multiple comparisons of factor contrasts. Specially suited for the analysis of interaction effects.

Details

Package:	phia
Type:	Package
Version:	0.3-2
Date:	2025-05-30
License:	GPL (>= 3)

This package contains functions that may be used for the post-hoc analysis of any term of linear models (univariate or multivariate), generalized and mixed linear models. The function `testFactors` provides a flexible user interface for defining combinations of factor levels and covariates, to evaluate and test the model, using the function `linearHypothesis` from package **car**. `testInteractions` uses this function for multiple comparisons of simple effects, interaction residuals, interaction contrasts, or user-defined contrasts. `interactionMeans` may be used to explore the 'cell means' of factorial designs, and plot main effects or first-order interactions.

Author(s)

Helios De Rosario-Martinez, with code contributions from John Fox (package **car**) and the R Core Team (**stats** package).

Maintainer: Helios De Rosario-Martinez <helios.derosario@gmail.com>

`Boik` *Contrived Data of Treatments for Hemophobia*

Description

Data set based on the hypothetical example used by R.J. Boik (1979) to explain the analysis of interaction contrasts. It represents the electrodermal response of 72 students complaining of hemophobia, treated with different fear reduction therapies and doses of antianxiety medication, in a balanced factorial design. The observed values of the dependent variable (not given in the original article) are contrived so that the results of all the tests are coherent with the examples.

Usage

```
Boik
```

Format

A data frame with 72 observations and three columns:

`therapy` Fear reduction therapy. Factor with levels `control`, `T1`, `T2`.

`medication` Dose of antianxiety medication. Ordered factor with levels `placebo`, `D1`, `D2`, `D3`.

`edr` Electrodermal response (in arbitrary units).

Note

The anova table in Boik's article (p. 1085) has a misprint in the MS value for 'Treatment B' (medication): it should be 790.32, instead of 970.32.

Source

Boik, R. J. (1979). 'Interactions, Partial Interactions, and Interaction Contrasts in the Analysis of Variance', *Psychological Bulletin*, 86(5), 1084-1089.

`contrastCoefficients` *Calculate Coefficient Matrices of Factor Contrasts*

Description

Take symbolic formulas of contrasts across the levels of one or more factors, and return a list of matrices with the corresponding linear combination coefficients, in a suitable form to use in `testFactors`, or as custom contrasts in `testInteractions`.

Usage

```
contrastCoefficients(..., contrast.definitions,
data=parent.frame(), normalize=FALSE)
```

Arguments

`..., contrast.definitions`	definitions of the contrasts.
`data`	list, data frame or environment that contains the factors symbolically represented in the contrast definitions.
`normalize`	logical value: should the coefficients be normalized to unit norm?

Details

In the context of this function, a "contrast" means a linear combination of factor levels (regardless of the dummy coefficients that are used to code those levels in fitted models). For a factor `f` with three levels `f1`, `f2`, `f3`, this could be a single level (e.g. `f ~ f1`), a "pairwise" contrast (e.g. `f ~ f1 - f2`), an average (e.g. `f ~ (f1 + f2 + f3) / 3`), or any other linear combination. Such arithmetic operations are usually applied to the values of a dependent variable, conditioned to the value of the represented factor, but those symbolic representations, as if the levels of the factors were themselves combined, are useful abstractions that may come in handy to define hypotheses about the effects of the factor.

This function takes one or more formulas of that type, and returns matrices of the linear coefficients that define such contrasts. For the previous examples, it would return the column matrices `list(f=matrix(c(f1=1, f2=0, f3=0)))`, `list(f=matrix(c(f1=1, f2=-1, f3=0)))`, and `list(f=matrix(c(f1=0.333, f2=-0.333, f3=0.333)))`, respectively. The factors must be defined in the data frame, list, or environment given in `data`, with the name provided in the right hand side of the formulas. By default this is the `parent.frame`, which in normal interactive use is the global environment.

The contrast matrices are returned in a named list, where the names are the represented factors, as required by the arguments `levels` and `custom` of the functions `testFactors` and `testInteractions`, respectively. When more than one formula is given for the same factor, all the corresponding columns of coefficients are bound into a single matrix.

Other ways of representing contrasts, allowed by `testFactors` and `testInteractions`, can also be mixed with the formulas as input arguments. Such alternative contrast definitions are just appended without changes in the returned list, after the matrices created from the formulas. (Notice that if other coefficient vectors or matrices are explicitly entered, these will not be combined with the ones calculated from formulas.)

In case of having the contrast definitions stored in a list, that list can be entered as the argument `contrast.definitions`. This option is recommended if the contrast definitions are mixed with named elements which could create a conflict with `data`, `normalize`, or `contrast.definitions` itself (although this is unlikely to happen specially if all the definitions are given as formulas). In any event, only one alternative of entering the definitions is allowed. If `contrast.definitions` is given, all the other definitions are ignored with a warning.

Value

A named list, where each element has the name of a factor, and contains a numeric matrix with the contrast coefficients. Each row of such matrices is associated to a level of the factor, in the order given by `levels`.

Author(s)

Helios De Rosario-Martinez, <helios.derosario@gmail.com>

See Also

`testFactors`, `interactionMeans`.

Examples

```
# Calculate the coefficients of example(testInteractions)
# cntrl.vs.T1 <- list(therapy = c(1, -1, 0))
contrastCoefficients(therapy ~ control - T1, data = Boik)
# cntrl.vs.T2 <- list(therapy = c(1, 0, -1))
contrastCoefficients(therapy ~ control - T2, data = Boik)
# plcb.vs.doses <- list(medication = c(1, -1/3, -1/3, -1/3))
contrastCoefficients(medication ~ placebo - (D1+D2+D3)/3, data = Boik)
# Combine cntrl.vs.T1 and plcb.vs.doses
contrastCoefficients(
  therapy ~ control - T1, medication ~ placebo - (D1+D2+D3)/3, data = Boik)

# Put various contrasts of the same factor in a matrix, and normalize them
contrastCoefficients(
therapy ~ control - T1,
therapy ~ control - T2,
medication ~ placebo - (D1+D2+D3)/3,
data = Boik,
normalize=TRUE)
```

`interactionMeans` *Calculate and Plot Adjusted Means for Interactions*

Description

Creates a data frame with the adjusted means of a fitted model or the slopes associated to its covariates, plus the standard error of those values, for all the interactions of given factors, including intra-subjects factors in multivariate linear models with intra-subjects designs. These interactions may be plotted by pairs of factors.

Usage

```
interactionMeans(model, factors=names(xlevels), slope=NULL, ...)
## S3 method for class 'interactionMeans'
plot(x, atx=attr(x,"factors"), traces=atx, xlab=atx,
 ylab="", main=NULL, multiple=TRUE, y.equal=FALSE, legend=TRUE, legend.margin=0.2,
  cex.legend=1, abbrev.levels=FALSE, type="b", pch=0:6, errorbar, ...)
```

Arguments

`model`	fitted model. Currently supported classes include `"lm"`, `"glm"`, `"mlm"`, `"lme"`, and `"mer"` or `"merMod"` (excluding models fitted by `nlmer`).
`factors`	character vector with the names of interacting factors. All the factors of the model are used by default; use `NULL` for calculating the overall mean.
`slope`	character vector with the names of the interacting covariates associated to the slope that will be represented; if it is `NULL` (the default value), the function will calculate adjusted mean values.

x	an object created by the function interactionMeans.
atx	character vector with the names of factors that will be represented in the horizontal axis of the plots. All the factors represented in x are used by default.
traces	character vector with the names of the factors that will be represented as different traces in a plot (the same as atx by default).
xlab, ylab, main	values to override the default titles of plots and their axes (see *Details*).
multiple	logical indicating if the plots shall be combined in multi-panel figures.
y.equal	logical indicating if all the plots should have the same range in the y-axis (all plots would be expanded to the maximum range).
legend	logical indicating if legends should be drawn in case of plots with multiple traces.
legend.margin	fraction of the combined plotting area that shall be reserved for drawing the legends, if they are required.
cex.legend	character expansion for the legend, relative to the value of par("cex").
abbrev.levels	A logical indicating whether the factor levels are to be abbreviated in the plot, or an integer that specifies the minimum length of such abbreviations. See abbreviate.
type, pch	type of plot and point characters, as used by matplot.
errorbar	definition of the error bars in the plots (see *Details*).
...	further arguments passed to testFactors or matplot.

Details

This function calculates the adjusted values of the model and their standard errors for interactions between factors, at fixed values of covariates, if they exist. The main or crossed effect of covariates is represented by their "slope", i.e. the variation rate of the response with respect to the product of the specified covariates. The default value of the covariates (and of the offset, if any) is their average in the model data frame, and it can be changed by the arguments covariates or offset, passed down to testFactors. Note that in generalized linear models, standard errors and slopes are referred to the link function, not to the mean (see testFactors for details, and how to force the calculation of the link function instead of the response for adjusted means).

In multivariate linear models, the adjusted means or slopes are calculated separately for each column by default, but it is possible to define an intra-subjects design of factors across columns, and put all columns in one. This may be defined by the argument idata passed down to testFactors (see Anova or linearHypothesis in package **car** for further details). If such transformation is done, it is also possible to include the factors of the intra-subjects design in factors, for calculating their main effects or interactions.

The generic plot function creates matrices of interaction plots, with the main effects of each factor represented in the diagonal, and the interactions between each pair of factors in the rest of panels. For multivariate models without intra-subjects design, a new device for each variable will be created. By default it also prints error bars around the means, plus/minus their standard errors. The size of the error bars can be adjusted by the argument errorbar. Currently supported definitions are strings with the pattern ciXX, where XX is a number between 01 and 99, standing for the $XX

The adjusted means and error bars of generalized models (fitted with `glm` or `glmer`) are plotted on the scale of the link function, although the y-axis is labelled on the scale of the response (unless the link function had been forced in the calculation of the means).

If the interactions involve many factors, it may be convenient to plot each panel in a different device (with `multiple=FALSE`), or select a subset of factors to be plotted with the arguments `atx` and `traces`. A panel will be plotted for each pair of factors defined by crossing these arguments; if the crossed factors are the same one, that panel will show its main effect.

By default, the interaction plots show the names of the factors on the labels of their x-axes (only for the bottom row of panels, if there is more than one). Those labels can be overriden by the argument `xlab`, which should be a character vector with as many strings as the number of factors of the model (or the number of factors indicated in `atx`). The string passed as `ylab` will be used as the label of the y-axes on the left column of panels (no label is printed by default). The default main titles of the plots are the names of the dependent variables (e.g. "adjusted mean"), but they can be also replaced by a string passed in the argument `main`, which will be used for all the variables if there is more than one.

Value

`interactionMeans` returns an object of class `"interactionMeans"`, that contains a data frame with the factor interactions and the means or slopes of the model adjusted to them, and some attributes used for plotting.

Note

The purpose of the `plot` method is similar to the function `interaction.plot`, but it uses the lower-level function `matplot`, so the aspect of the plots is different.

Author(s)

Helios De Rosario-Martinez, <helios.derosario@gmail.com>

See Also

`testFactors`, `interactionMeans`.

Examples

```
# Interaction between two factors
# See ?Adler for a description of the data set

mod.adler <- lm(rating ~ instruction * expectation, data=Adler)
(means.adler <- interactionMeans(mod.adler))
plot(means.adler, abbrev.levels=TRUE)


# Effect of factors on the slopes of the model
# See ?SLID for a description of the data set

SLID$logwages <- log2(SLID$wages)
mod.slid <- lm(logwages ~ education + age * (sex  * language), data=SLID)
```

```
(slopes.slid <- interactionMeans(mod.slid, slope="age"))
plot(slopes.slid)


# Include intra-subjects factors
# See ?OBrienKaiser for a description of the data set

mod.ok <- lm(cbind(pre.1, pre.2, pre.3, pre.4, pre.5,
                   post.1, post.2, post.3, post.4, post.5,
                   fup.1, fup.2, fup.3, fup.4, fup.5) ~  treatment*gender,
               data=OBrienKaiser)

# Intra-subjects data:
phase <- factor(rep(c("pretest", "posttest", "followup"), each=5))
hour <- ordered(rep(1:5, 3))
idata <- data.frame(phase, hour)

# Calculate all interactions, but plot only the interactions between
# hour (represented in the x-axis) and the other factors (in traces)

means.ok <- interactionMeans(mod.ok, idata=idata)
plot(means.ok, atx="hour", traces=c("gender","treatment","phase"))
```

`Keselman` *Repeated-Measures Pyschopsychological Experiment*

Description

This data set represents the outcome of an hypothetical experiment, that recorded psychophysiological measures of subjects with different susceptibility to stressors. Each subject performed a task at four different levels of challenge. Data artificially generated from a multivariate lognormal distribution, with unequal variances and covariance structure related to group sample sizes.

Usage

```
Keselman1
Keselman2
```

Format

`Keselman1` is a data frame of 156 rows and 4 columns, with the measures of 39 subjects in a balanced design.

`Keselman2` is an unbalanced subset of `Keselman1`, with 120 rows and 4 columns, corresponding to the measures of 30 subjects.

Both data frames contain the following variables:

`subject` Integer identity of the subject.

`group` Classification of the subjects according to their susceptbility to stressors. Factor with levels `G1`, `G2`, and `G3`.

`challenge` Level of challenge of the task. Ordered factor with levels `M1`, `M2`, `M3`, `M4`.

`measure` Psychophisiological measure.

Source

Keselman, H. J. (1998) 'Testing treatment effects in repeated measures designs: an update for psychophysiological researchers'. *Psychophysiology*, 35(4), 470-478.

`Rosnow` *Rosnow's and Rosenthal's Baseball Performance Data*

Description

Contrived data set of an hypothetical study about the efficacy of a new training method for improving the performance of baseball players. 2x2 factorial design with two factors: training method and expertise of the player.

Usage

```
Rosnow
```

Format

A data frame with 36 observations and three columns:

`method` Training method. Factor with levels `control` and `Ralphing`.

`experience` Experience of the player in championship competition. Factor with levels `no`, `yes`.

`hits` Performance scores, measured as total number of hits in an experimentally devised game.

Source

Rosnow, R. L., & Rosenthal, R. (1989). 'Definition and Interpretation of Interaction Effects', *Psychological Bulletin*, 105(1), 143-146.

testFactors *Evaluate and Test Combinations of Factor Levels*

Description

Calculates and tests the adjusted mean value of the response and other terms of a fitted model, for specific linear combinations of factor levels and specific values of the covariates. This function is specially indicated for post-hoc analyses of models with factors, to test pairwise comparisons of factor levels, simple main effects, and interaction contrasts. In multivariate models, it is possible to define and test contrasts of intra-subjects factors, as in a repeated-measures analysis.

Usage

```
## Default S3 method:
testFactors(model, levels, covariates, offset, terms.formula=~1,
  inherit.contrasts=FALSE,  default.contrasts=c("contr.sum","contr.poly"),
  lht=TRUE, ...)
## S3 method for class 'lm'
testFactors(model, ...)
## S3 method for class 'glm'
testFactors(model, ..., link=FALSE)
## S3 method for class 'mlm'
testFactors(model, levels, covariates, offset, terms.formula=~1,
  inherit.contrasts=FALSE, default.contrasts=c("contr.sum","contr.poly"),
  idata, icontrasts=default.contrasts, lht=TRUE, ...)
## S3 method for class 'lme'
testFactors(model, ...)
## S3 method for class 'mer'
testFactors(model, ..., link=FALSE)
## S3 method for class 'merMod'
testFactors(model, ..., link=FALSE)
## S3 method for class 'testFactors'
summary(object, predictors=TRUE, matrices=TRUE, covmat=FALSE, ...)
```

Arguments

model	fitted model. Currently supported classes include "lm", "glm", "mlm", "lme", and "mer" or "merMod" (excluding models fitted by nlmer).
levels	list that defines the factor levels that will be contrasted; see the *Details* for more information.
covariates	optional numeric vector with specific values of the covariates (see *Details*).
offset	optional numeric value with a specific value of the offset (if any).
terms.formula	formula that defines the terms of the model that will be calculated and tested. The default value is ~1, and stands for the adjusted mean value. See the *Details* for more information.

`inherit.contrasts`
: logical value: should the default contrasts of model factors be inherited from the model data frame?

`default.contrasts`
: names of contrast-generating functions to be applied by default to factors and ordered factors, respectively, if `inherit.contrasts` is `FALSE` (the default); the contrasts must produce an intra-subjects model matrix in which different terms are orthogonal. The default is `c("contr.sum", "contr.poly")`.

`lht`
: logical indicating if the adjusted values are tested (via `linearHypothesis`).

`link`
: for models fitted with `glm` or `glmer`, logical indicating if the adjusted mean values should represent the link function (`FALSE` by default, i.e. represent the adjusted means of the response variable).

`idata`
: an optional data frame giving a factor or factors defining the intra-subjects model for multivariate repeated-measures data, as defined in `Anova` or `linearHypothesis`.

`icontrasts`
: names of contrast-generating functions to be applied in the within-subject "data". The default is the same as `default.contrasts`.

`object`
: object returned by `testFactors`.

`predictors`
: logical value: should `summary` return the values of the predictor values used in the calculations?

`matrices`
: logical value: should `summary` return the matrices used for testing by `linearHypothesis`?

`covmat`
: logical value: should `summary` return the covariance matrix of the adjusted values?

`...`
: other arguments passed down to `linearHypothesis`.

Details

The only mandatory argument is `model`, which may include any number of factor or numeric predictors, and one offset. The simplest usage of this method, where no other argument is defined, calculates the adjusted mean of the model response variable, pooling over all the levels of factor predictors, and setting the numeric predictors (covariates and offset, ifany) to their average values in the model data frame.

The calculations will be done for the linear combinations of factor levels defined by `levels`. This argument must be a list, with one element for each factor of the model that has to be manipulated (including factors of the intra-subjects design, if suitable). The factors that are not represented in this list will be pooled over, and elements that do not correspond to any factor of the model will be ignored with a warning. `levels` may be a named list, where the name of each element identifies the represented factor, and its contents may be one of the following:

1. A character string of length 1 or 2, with the name of one or two factor levels. In the former case, the calculations will be restricted to this level of the factor, as in a simple main effects analysis; in the latter, a pairwise contrast will be calculated between both factor levels.
2. A numeric vector without names, as long as the number of levels in the factor. This will create a linear combination of factor levels, with the elements of the vector as coefficients. For instance, if the factor `f` has three levels, an element `f=c(0.5, 0.5, 0)` will average the two first levels, and `f=c(0.5, 0.5, -1)` will contrast the average of the two first levels against the third one.

3. A numeric vector with names equal to some or all the levels of the factor. This is a simplification of the previous option, where some levels can be omitted, and the coefficient of each level is determined by the names of the vector, which do not have to follow a specific order. Omitted levels will automatically be set to zero.
4. A numeric matrix, as an extension of the two previous options for calculating several combinations at a time. Combinations are defined in columns, so if the matrix does not have row names, the number of rows must be equal to the number of levels in the factor, or if the matrix does have row names, they must coincide with the levels of the factor.

Alternatively, `levels` may be a single formula or an unnamed list of formulas, of the type `factorname ~ K1*level1 + K2*level2 ...` (see `contrastCoefficients` for further details). Both types of lists (named list of string or numeric vectors and matrices, and unnamed lists of formulas) may be mixed.

The argument `covariates` may be used for setting specific values of the model numeric predictors (covariates). It must be a vector of numeric values. If the elements of this vector have names, their values will be assigned to the covariates that match them; covariates of the model with names not represented in this vector will be set to their default value (the average in the model data frame), and elements with names that do not match with covariates will be ignored. On the other hand, if `covariates` has no names, and its length is equal to the number of covariates of the model, the values will be assigned to those covariates in the same order as they occur in the model. If it has a different length, the vector will be trimmed or reclycled as needed to fit the number of covariates in the model; this feature may be used, for instance, to set all covariates to the same value, e.g. `covariates = 0`. The argument `offset` can likewise be used to define a specific value for the offset of the model.

To analyse terms other than the adjusted mean value of the response, use the argument `terms.formula`. For instance, if the model has the covariates `var1`, `var2`, ..., the slopes of the response with respect to them may be added to the analysis by defining `terms.formula` as `~var1 + var2 ...`. This formula may be used more generally, for analysing interactions, omitting the mean response, adding the main effects of factors, etc. A different analysis is done for each term of this formula, that must also be contained in the formula of `model`. For instance, if `terms.formula` is equal to `~ var1*var2`, the function will analyse the adjusted intercept, plus the terms `var1`, `var2`, and `var1:var2`. The intercept stands for the mean value of the response, and terms formed by one or more covariates stand por the slope of the response with respect to the product of those covariates.

If any of the variables in the term is a factor, the function analyses a full set of contrasts for that factor of the remaining part of the term; for instance if `var1` were a factor, the term `var1` would stand for the contrasts of the intercept, and `var1:var2` would stand for the contrasts of the slope `var2`, across the levels of `var1`. The set of contrasts used in the analysis is normally defined by the argument `default.contrasts`: by default, if the factor is ordered it will be a set of "polynomial contrasts", and otherwise "sum contrasts"; however, if `inherit.contrasts` is `TRUE` the contrasts will directly be copied from the ones used to define the model. Factors that have explicit contrasts defined in the model data frame will use those contrasts, regardless of the values defined for `default.contrasts` and `inherit.contrasts`. The analysis assumes that the contrasts are orthogonal to the intercept, which is the usual case if the default arguments are provided, and a warning will be issued if non-orthogonal contrasts are used; take special care of not using "treatment contrats" if `inherit.contrasts` is set to `TRUE` or `default.contrasts` is changed.

In generalized linear models, the adjusted means represent the expected values of the response by default, but the expected value of the link function may be shown by setting the argument `link=FALSE`. On the other hand, slope values and standard errors always refer to the link function.

For multivariate models, the arguments idata, and icontrasts may be used to define an intra-subjects model for multivariate repeated-measures data, as described for Anova or linearHypothesis in package **car**. Note, however, that the combinations of intra-subjects factor levels are defined in levels, and other arguments defined in those functions like idesign, imatrix or iterms will have no effect in testFactors.

The significance of adjusted values is tested by a call to linearHypothesis for each term, unless lht is set to FALSE. Extra arguments may be passed down to that function, for instance to specify the test statistic that will be evaluated.

Value

An object of class "testFactors", that contains the adjusted values and their standard errors for each term, and the otuput of the test, plus other variables used in the calculations. The summary method for this object will display those variables, unless they be omitted by setting the optional arguments predictors, matrices or covmat to FALSE. The argument predictors refers to the coefficients of specified combinations of factor levels, the values of covariates, and the contrast matrices used for terms that include factors; matrices refers to the "linear hypothesis matrix" used by linearHypothesis, and in multivariate linear models, to the "response transformation matrix" as well — if it exists; covmat refers to the variance-covariance matrix of the adjusted values.

Moreover, summary groups the results of the tests for all terms in one table. By default this table shows the test statistics, their degrees of freedom, and the *p*-values. If the model is of class "lm", it also shows the sums of squares; and if it is of class "mlm", only the first type of test statistic returned by linearHypothesis (by default "Pillai") is shown. This variable shape of the ANOVA table is controlled by additional classes assigned to the object (either "testFactors.lm" or "testFactors.mlm", as suitable).

Note

The tests of mixed models are done under the assumption that the estimation of the random part of the model is exact.

Author(s)

Helios De Rosario-Martinez, <helios.derosario@gmail.com>

See Also

linearHypothesis in package **car**. interactionMeans, and testInteractions as useful wrappers of testFactors.

Examples

```
# Example with factors and covariates
# Analyse prestige of Canadian occupations depending on
# education, income and type of occupation
# See ?Prestige for a description of the data set

prestige.mod <- lm(prestige ~ (education+log2(income))*type, data=Prestige)

# Pairwise comparisons for factor "type", to see how it influences
```

```
# the mean value of  prestige and interacts with log.income

# 1: "white collar" vs "blue collar"
wc.vs.bc <- list(type=c("wc", "bc"))
testFactors(prestige.mod, wc.vs.bc, terms.formula=~log2(income))
# 2: "professional" vs. "blue collar"
prof.vs.bc <- list(type=c("prof", "bc"))
testFactors(prestige.mod, prof.vs.bc, terms.formula=~log2(income))
# 3: "professional" vs. "white collar"
prof.vs.wc <- list(type=c("prof", "wc"))
testFactors(prestige.mod, prof.vs.wc, terms.formula=~log2(income))


# Interaction contrasts in a repeated-measures experiment
# See ?OBrienKaiser for a description of the data set

mod.ok <- lm(cbind(pre.1, pre.2, pre.3, pre.4, pre.5,
  post.1, post.2, post.3, post.4, post.5,
  fup.1, fup.2, fup.3, fup.4, fup.5) ~  treatment*gender,
  data=OBrienKaiser)

# intra-subjects data:
phase <- factor(rep(c("pretest", "posttest", "followup"), each=5))
hour <- ordered(rep(1:5, 3))
idata <- data.frame(phase, hour)
Anova(mod.ok, idata=idata, idesign=~phase*hour)

# Contrasts across "phase", for different contrasts of "treatment"
# Using different definitions of the argument "levels"

# 1: Treatment "A" vs. treatment "B".
A.vs.B <- list(treatment=c("A", "B"))
# The following are equivalent:
# A.vs.B <- list(treatment=c(A=1, B=-1, control=0))
# A.vs.B <- list(treatment=c(A=1, B=-1))
# A.vs.B <- list(treatment ~ A - B)
# A.vs.B <- treatment ~ A - B
testFactors(mod.ok, A.vs.B, idata=idata, terms.formula=~0+phase)

# 2: Controls vs. treatments
control.vs.AB <- list(treatment=c(A=0.5, B=0.5, control=-1))
# The following is equivalent:
# control.vs.AB <- treatment ~ (A+B)/2 - control
testFactors(mod.ok, control.vs.AB, idata=idata, terms.formula=~0+phase)

# Shortcut to get only the adjusted values and simplified ANOVA tables
contr <- list(A.vs.B=A.vs.B, control.vs.AB=control.vs.AB)
anovaTables <- function(contrast) summary(testFactors(mod.ok, contrast,
  idata=idata, terms.formula=~0+phase),
  predictors=FALSE, matrices=FALSE)

lapply(contr,anovaTables)
```

testInteractions *Test Contrasts of Factor Interactions*

Description

Calculates and tests different types of contrasts for factor interactions, in linear, generalized and mixed linear models: simple main effects, interaction contrasts, residual effects, and others.

Usage

```
testInteractions(model, pairwise=NULL, fixed=NULL, residual=NULL, across=NULL,
  custom=NULL, slope=NULL, adjustment=NULL, label.factors=FALSE,
  abbrev.levels=FALSE, ...)
```

Arguments

model	fitted model. Currently supported classes include "lm", "glm", "mlm", "lme", and "mer" or "merMod" (excluding models fitted by nlmer).
pairwise	character vector with the names of factors represented by pairwise contrasts.
fixed	character vector with the names of factors represented by fixed levels.
residual	character vector with the names of factors represented by residuals effects.
across	character vector with the names of factors represented by a full set of independent contrasts.
custom	list with custom contrasts for other factors. See the Details for more information.
slope	character vector with the names of the covariates associated to the slope that will tested; if it is NULL (the default value), the function will test the adjusted mean values.
adjustment	adjustment method for p-values, as defined in p.adjust.
label.factors	If true, the rownames for each row in the resulting table include the name(s) of the factor(s) involved, followed by the level values. Otherwise, the rownames include only the levels of the factor(s), with multiple factors separated by ':'.
abbrev.levels	Either a logical or an integer, specifying whether the levels values of the factors in the term are to be abbreviated in constructing the rownames. An integer specifies the minimum length of the abbreviation for each factor in the term.
...	further arguments passed down to testFactors.

Details

Each factor of the model can at most be contained in one of the arguments pairwise, fixed, residual, across, or custom; redundant assignment of factors is not allowed. If none of these arguments is defined, the default behavior is as if pairwise contained all the factors of the model. The result will show a set of tests on the model adjusted mean, at different combinations of factor levels. If there are covariates defined in slope, the test will apply to the slope for the interaction of such covariates. Each row will contain a different combination of factor levels or contrasts, depending on the argument wherein the factor has been defined:

- The factors contained in `pairwise` will appear as pairwise contrasts between levels.
- The factors contained in `fixed` will appear as one of their possible levels.
- The factors contained in `residual` will appear as residual effects of their levels, after removing effects of higher order.
- The factors contained in `across` will appear as a full set of contrasts. By default they will be orthogonal contrasts, unless overriden by the contrasts of the model data frame or by the arguments passed down to `testFactors`. See the documentation of that function for further details.

Ommitted factors will be averaged across all their levels. Thus, to test the overall adjusted means or slopes, use `pairwise=NULL` (or do the same with any of the arguments of the previous list).

Other combinations of factor levels can be defined by `custom`. This argument should be a list of numeric matrices or vectors, named as the model factors. Each matrix must have as many rows as the number of levels of the corresponding factor, so that each column represents a linear combination of such levels that will be tested, crossed with the combinations of the other factors. Vectors will be treated as column matrices.

In multivariate linear models it is possible to define an intra-subjects design, with the argument `idata` passed down to `testFactors` (see `Anova` or `linearHypothesis` in package **car** for further details). The factors defined by that argument can be included as any other factor of the model.

Value

An anova table with one row for each different combination of levels and contrasts defined in `pairwise`, `fixed`, `across`, and `custom`. The rownames represent the specific levels or contrasts used for the different factors, separated by ':'. These names can be tweaked by the arguments `label.factors` and `abbrev.levels`, as done by `termMeans` in package **heplots**.

Note

The tests of mixed models are done under the assumption that the estimation of the random part of the model is exact.

Author(s)

Helios De Rosario-Martinez, <helios.derosario@gmail.com>

See Also

`testFactors`, `interactionMeans`. Use `contrastCoefficients` as a facility to create matrices of custom contrasts.

Examples

```
# Tests of the interactions described in Boik (1979)
# See ?Boik for a description of the data set

mod.boik <- lm(edr ~ therapy * medication, data=Boik)
Anova(mod.boik)
cntrl.vs.T1 <- list(therapy = c(1, -1, 0))
```

```
cntrl.vs.T2 <- list(therapy = c(1, 0, -1))
plcb.vs.doses <- list(medication = c(1, -1/3, -1/3, -1/3))
testInteractions(mod.boik, pairwise="therapy", adjustment="none")
testInteractions(mod.boik, custom=plcb.vs.doses, adjustment="none")
testInteractions(mod.boik, custom=cntrl.vs.T1, across="medication", adjustment="none")
testInteractions(mod.boik, custom=c(cntrl.vs.T1, plcb.vs.doses), adjustment="none")
testInteractions(mod.boik, custom=cntrl.vs.T2, across="medication", adjustment="none")
testInteractions(mod.boik, custom=plcb.vs.doses, across="therapy", adjustment="none")
```

Index