Package 'lsmeans'

July 26, 2015

Type Package

Title Least-Squares Means

Version 2.19

Date 2015-07-26

Encoding latin1

Author Russell V. Lenth [aut], Maxime Hervé [ctb]

Maintainer Russ Lenth <russell-lenth@uiowa.edu>

Depends estimability, methods, R (>= 3.0)

Suggests car, lattice, MCMCpack, mediation, multcompView, ordinal, pbkrtest (>= 0.4-1), afex, CARBayes, coxme, gee, geepack, glmmADMB, lme4.0, lme4, MASS, MCMCglmm, survival

Imports graphics, stats, utils, nlme, coda (>= 0.17), multcomp, plyr, mvtnorm

Additional_repositories http://glmmadmb.r-forge.r-project.org/repos, http://lme4.0.r-forge.r-project.org/repos

LazyData yes

ByteCompile yes

Description Obtain least-squares means for many linear, generalized linear, and mixed models. Compute contrasts or linear functions of least-squares means, and comparisons of slopes. Plots and compact letter displays.

License GPL-2

NeedsCompilation no

Repository CRAN

Date/Publication 2015-07-26 22:51:43

R topics documented:

1

lsmeans-package *Least-squares means*

Description

This package provides methods for obtaining so-called least-squares means for factor combinations in a variety of fitted linear models. It can also compute contrasts or linear combinations of these least-squares means, (several standard contrast families are provided), and in addition can estimate and contrast slopes of trend lines. Popular adjustments for multiple-comparisons are provided, as well as graphical ways of displaying the results.

Details

Package:	lsmeans
Type:	Package
License:	GPL-2
Other information:	See DESCRIPTION

Overview

Concept Least-squares means (see Searle *et al.* 1980, who prefer the term "predicted marginal means" (PMM)) are popular for summarizing linear models that include factors. For balanced experimental designs, they are just the marginal means. For unbalanced data, they in essence estimate what you *would* have observed that the data arisen from a balanced experiment.

Reference grids The implementation in **lsmeans** relies on our own concept of a *reference grid*, which is an array of factor and predictor levels. Predictions are made on this grid, and least-squares means are defined as averages of these predictions over zero or more dimensions of the grid. The function `ref.grid` explicitly creates a reference grid (`ref.grid` object) that can subsequently be used to obtain least-squares means. The `update` method is used to change its properties.

Our reference-grid framework expands slightly upon Searle *et al.*'s definitions of PMMs, in that it is possible to include multiple levels of covariates in the grid.

Models supported Many linear models are supported by the package, including `lm`, `glm`, `aovList`, and `mlm` in the **stats** package, as well as fitted-model objects from several contributed packages including **nlme**, **lme4**, **survival**, and **geepack**. The help page for `models` provides more details, including, in some cases, additional `ref.grid` arguments that might affect the subsequent analysis. Also, some models require other packages be installed in order to obtain all the available features.

Least-squares means The `lsmeans` function computes least-squares means given a `ref.grid` object or a fitted model, and a specification indicating what factors to include. The `lstrends` function creates the same sort of results for estimating and comparing slopes of fitted lines. Both return an `lsmobj` object very much like a reference grid, but with possibly fewer factors involved.

Summaries and analysis The `summary` method may be used to display a `ref.grid` or an `lsmobj`. Special-purpose summaries are available via `confint` and `test`, the latter of which can also do a joint test of several estimates. The user may specify by variables, multiplicity-adjustment methods, confidence levels, etc., and if a transformation or link function is involved, may reverse-transform the results to the response scale.

Contrasts and comparisons The `contrast` method is used to obtain contrasts among the estimates; several standard contrast families are available such as deviations from the mean, polynomial contrasts, and comparisons with one or more controls. Another `lsmobj` object is returned, which can be summarized or further analyzed. For convenience, a `pairs` method is provided for the case of pairwise comparisons. Related to this is the `cld` method, which provides a compact letter display for grouping pairs of means that are not significantly different. `cld` requires the **multcompView** package.

Graphs The `plot` method will display side-by-side confidence intervals for the estimates, and/or 'comparison arrows' whereby the significance of pairwise differences can be judged by how much they overlap. The `lsmip` function displays estimates like an interaction plot, multi-paneled if there are by variables. These graphics capabilities require the **lattice** package be installed.

multcomp interface The `as.glht` function and `glht` method for `lsmobj`s provide an interface to the `glht` function in the **multcomp** package, thus providing for more exacting simultaneous estimation or testing. The package also provides an `lsm` method that works as an alternative to `mcp` in a call to `glht`.

Additional information

Examples and discussion are available via `vignette("using-lsmeans", package="lsmeans")`.

Some features of the **lsmeans** require (or are enhanced by) additional packages that are loaded when needed. Since they are not "required" packages, they are not automatically installed with **lsmeans**.

We highly recommend that users also install the following packages: **multcomp** (if `cld`, `glht`, or `as.glht` are to be used), **multcompView** (for `cld`), **lattice** (for `plot` and `lsmip`), and **pbkrtest** (for models fitted by the **lme4** package).

Starting with **lsmeans** version 2, a new object framework based on *reference grids* is used that increases flexibility and provides for extending its capabilities to additional model objects. Use `vignette("lsmeans-changes")` for information on the user impact of these changes.

It is possible to write your own interfaces for models not yet supported by **lsmeans**. See the help page `extending-lsmeans` and `vignette("extending")` for details on how to do this.

Author(s)

Russell V. Lenth (author), Maxime HervÃ© (contributor)

Maintainer: Russ Lenth <russell-lenth@uiowa.edu>

References

Searle S.R. Speed F.M. Milliken G.A. (1980) Population marginal means in the linear model: An alternative to least squares means. *The American Statistician* **34**(4), 216-221.

---

auto.noise	*Auto Pollution Filter Noise*

Description

Three-factor experiment comparing pollution-filter noise for two filters, three sizes of cars, and two sides of the car.

Usage

```
auto.noise
```

Format

A data frame with 36 observations on the following 4 variables.

noise Noise level in decibels - a numeric vector.

size The size of the vehicle - an ordered factor with levels S, M, L.

type Type of anti-pollution filter - a factor with levels Std and Octel

side The side of the car where measurement was taken – a factor with levels L and R.

Details

The data are from a statement by Texaco, Inc., to the Air and Water Pollution Subcommittee of the Senate Public Works Committee on June 26, 1973. Mr. John McKinley, President of Texaco, cited an automobile filter developed by Associated Octel Company as effective in reducing pollution. However, questions had been raised about the effects of filters on vehicle performance, fuel consumption, exhaust gas back pressure, and silencing. On the last question, he referred to the data included here as evidence that the silencing properties of the Octel filter were at least equal to those of standard silencers.

Source

The dataset was imported from the Data and Story Library - [http://lib.stat.cmu.edu/DASL/](http://lib.stat.cmu.edu/DASL/Datafiles/airpullutionfiltersdat.html) [Datafiles/airpullutionfiltersdat.html](http://lib.stat.cmu.edu/DASL/Datafiles/airpullutionfiltersdat.html) (sic). However, the factor levels were assigned meaningful names, and the observations were sorted in random order as if this were the run order of the experiment.

References

A.Y. Lewin and M.F. Shakun (1976) *Policy Sciences: Methodology and Cases.* Pergammon Press. p.313.

Examples

```
require(lsmeans)
noise.lm <- lm(noise ~ size * type * side, data = auto.noise)

# Interaction plot of predictions
lsmip(noise.lm, type ~ size | side)

# Confidence intervals
plot(lsmeans(noise.lm, ~ size | side*type))
```

---

cld   *Compact letter display of pairwise comparisons*

Description

Extract and display information on all pairwise comparisons of least-squares means.

Usage

```
## S3 method for class 'ref.grid'
cld(object, details = FALSE, sort = TRUE, by, alpha = 0.05,
    Letters = c("1234567890", LETTERS, letters), ...)
## S3 method for class 'lsm.list'
cld(object, ..., which = 1)
```

Arguments

object	An object of class `ref.grid`
details	Logical value determining whether detailed information on tests of pairwise comparisons is displayed
sort	Logical value determining whether the LS means are sorted before the comparisons are produced
by	Character value giving the name or names of variables by which separate families of comparisons are tested. If `NULL`, all means are compared. If missing, and a by variable was used in creating `object`, it is used as the by variable in `cld`.

alpha	Numeric value giving the significance level for the comparisons
Letters	Character vector of letters to use in the display. Any strings of length greater than 1 are expanded into individual characters
...	Arguments passed to [contrast](for example, an adjust method)
which	When object is a list, this determines which element is analyzed.

Details

This function uses the Piepho (2004) algorithm (as implemented in the **multcompView** package) to generate a compact letter display of all pairwise comparisons of least-squares means. The function obtains (possibly adjusted) *P* values for all pairwise comparisons of means, using the [contrast](function with method = "pairwise". When a P value exceeds alpha, then the two means have at least one letter in common.

Value

When details == FALSE, an object of class summary.ref.grid (which inherits from data.frame) showing the summary of LS means with an added column named .groups with the cld information. When details == TRUE, a list the object just described, as well as the summary of the contrast results showing each comparison, its estimate, standard error, *t* ratio, and adjusted *P* value.

Note

This function requires the **multcompView** package to be installed. Otherwise an error message is produced.

Author(s)

Russell V. Lenth

References

Hans-Peter Piepho (2004) An algorithm for a letter-based representation of all pairwise comparisons, *Journal of Computational and Graphical Statistics*, 13(2), 456-466.

See Also

[cld](in the **multcomp** package

Examples

```
warp.lm <- lm(breaks ~ wool * tension, data = warpbreaks)
warp.lsm <- lsmeans(warp.lm, ~ tension | wool)
cld(warp.lsm)                    # implicitly uses by = "wool"
cld(warp.lsm, by = "tension")  # overrides implicit 'by'

# Mimic grouping bars and compare all 6 means
cld(warp.lsm, by = NULL, Letters = "|||||||||", alpha = .01)
```

---

contrast   *Methods for obtaining analyses* ref.grid *and* lsmobj *objects*

Description

These methods provide for analyses of ref.grid objects, or follow-up analyses of lsmobj objects: Contrasts, pairwise comparisons, tests, and confidence intervals.

Usage

```
## S3 method for class 'ref.grid'
contrast(object, method = "eff", by, adjust,
    offset = NULL, name = "contrast", options = getOption("lsmeans")$contrast, ...)
## S3 method for class 'lsm.list'
contrast(object, ..., which = 1)

## S3 method for class 'ref.grid'
test(object, null = 0, joint = FALSE,
     verbose = FALSE, rows, by,  ...)

## S3 method for class 'ref.grid'
confint(object, parm, level = 0.95, ...)

## S3 method for class 'ref.grid'
pairs(x, reverse = FALSE, ...)
```

Arguments

object, x	An object of class "ref.grid" or its extension, "lsmobj".
method	Character value giving the root name of a contast method (e.g. "pairwise"). Alternatively, a named list of contrast coefficients that must each conform to the number of least-squares means in each by group. This is just like the contr argument in [lsmeans](). To identify the available methods, see
	ls("package:lsmeans", pat=".lsmc")
	You may define your own .lsmc function and use its root name as method.
by	Character names of variable(s) to be used for "by" groups. The contrasts or joint tests will be evaluated separately for each combination of these variables. If object was created with by groups, those are used unless overridden. Use by = NULL to use no by groups at all.
adjust	Method to use for adjusting *P* values. This is passed to [summary]().
offset	Numeric vector of the same length as each by group. These values are added to their respective linear estimates.
name	Name to use to label the contrasts in table headings or subsequent contrasts of the returned object.

options	If non-NULL, a named `list` of arguments to pass to update, just after the object is constructed.
joint	Logical value. If `FALSE`, the arguments are passed to summary with `infer=c(FALSE,TRUE)`. If `TRUE`, a joint test of the hypothesis L beta = null is performed, where L is `object@linfct` and beta is the vector of fixed effects estimated by `object@betahat`. This will be either an *F* test or a chi-square (Wald) test depending on whether degrees of freedom are available.
rows	Integer values. The rows of L to be tested in the joint test. If missing, all rows of L are used. If not missing, by variables are ignored.
null	Numeric value specifying the null value(s) being tested against. It may be either a single value, in which case it is used as the null value for all linear functions under test; or a numeric vector of length equal to the number of linear functions.
parm	This is ignored, but it is a required argument of the generic `confint` method.)
verbose	Logical value. If `TRUE` and `joint==TRUE`, a table of the effects being tested is printed.
level	Numeric value of the desired confidence level.
which	When `object` is a list of `lsmobj` objects, this specifies which member of the list is analyzed.
reverse	Logical value determining whether `"pairwise"` or `"revpairwise"` pairwise comparisons are generated.
...	Additional arguments passed to summary or to a contrast function.

Details

Though `contrast` is ordinarily used to create true contrasts (whose coefficients sum to zero), it may be used to estimate any linear function of the LS means; and `offset` expands this capability further by allowing additive constants. `pairs` is equivalent to `contrast` with `method = "pairwise"`.

`confint` and `test` (when JOINT==FALSE) are equivalent to calling summary with `infer=c(TRUE,FALSE)` and `infer=c(FALSE,TRUE)`, respectively.

When using `test` to do a joint test of L beta = null, an error is thrown if any row of L is non-estimable. It is permissible for the rows of L to be linearly dependent as long as `null == 0`; a reduced set of contrasts is tested. Linear dependence and nonzero `null` cause an error.

Value

`contrast` and `pairs` return an object of class `"lsmobj"`, which is an extension of `"ref.grid"`. Consequently, they may be used as arguments to other `"lsmobj"` or `"ref.grid"` methods. The user may, for example, compute contrasts of contrasts, or re-summarize a set of confidence intervals with a different by grouping or confidence level. The "grid" for the returned value is simply the set of variables that identify the results. For example, `contrast`'s return value is a reference grid for one factor named `contrast`.

`confint` and `test` (when Joint==FALSE) return an object of class `summary.ref.grid`. When JOINT==TRUE, `test` returns a numeric vector with the test statistic, degrees of freedom, and *P* value.

Author(s)

Russell V. Lenth

See Also

Additional ″lsmobj″ methods having their own help pages are [cld](#) and [glht](#). Also, the [summary](#) and other methods for ″ref.grid″ objects also work for ″lsmobj″ objects.

Examples

```
require(lsmeans)
warp.lm <- lm(breaks ~ wool*tension, data = warpbreaks)
warp.lsm <- lsmeans(warp.lm, ~ tension | wool)

# Polynomial contrasts of tension, by wool
(warp.pl <- contrast(warp.lsm, ″poly″, name = ″order″))
# Same results with a different adjustment
summary(warp.pl, adjust = ″fdr″)

# Jointly test the tension effects for each wool
test(warp.pl, joint = TRUE, by = ″wool″)

# Compare the two contrasts for each order
contrast(warp.pl, ″revpairwise″, by = ″order″)

# User-provided contrasts, ignoring the previous by grouping
contrast(warp.lsm,
    list(c1=c(1,0,0,-1,0,0), c2=c(1,1,1,-1,-1,-1)/3),
    by = NULL)
```

---

feedlot   *Feedlot data*

Description

This is an unbalanced analysis-of-covariance example, where one covariate is affected by a factor. Feeder calves from various herds enter a feedlot, where they are fed one of three diets. The weight of the animal at entry is the covariate, and the weight at slaughter is the response.

Usage

```
data(feedlot)
```

Format

A data frame with 67 observations on the following 4 variables.

herd a factor with levels 9 16 3 32 24 31 19 36 34 35 33, designating the herd that a feeder calf came from.

diet a factor with levels Low Medium High: the energy level of the diet given the animal.

swt a numeric vector: the weight of the animal at slaughter.

ewt a numeric vector: the weight of the animal at entry to the feedlot.

Details

The data arise from a Western Regional Research Project conducted at New Mexico State University. Calves born in 1975 in commercial herds entered a feedlot as yearlings. Both diets and herds are of interest as factors. The covariate, ewt, is thought to be dependent on herd due to different genetic backgrounds, breeding history, etc. The levels of herd ordered to similarity of genetic background.

Note: There are some empty cells in the cross-classification of herd and diet.

Source

Urquhart NS (1982) Adjustment in covariates when one factor affects the covariate. *Biometrics* 38, 651-660.

Examples

```
require(lsmeans)
feedlot.lm <- lm(swt ~ ewt + herd*diet, data = feedlot)

# Obtain LS~means with a separate reference value of ewt for each
# herd. This reproduces the last part of Table 2 in the reference
lsmeans(feedlot.lm,  ~ diet | herd,  cov.reduce = ewt ~ herd)
```

---

fiber   *Fiber data*

Description

Fiber data from Montgomery Design (8th ed.), p.656 (Table 15.10). Useful as a simple analysis-of-covariance example.

Usage

```
fiber
```

Format

A data frame with 15 observations on the following 3 variables.

machine a factor with levels A B C. The primary factor of interest.

strength a numeric vector. The response variable.

diameter a numeric vector. A covariate.

Details

The goal of the experiment is to compare the mean breaking strength of fibers produced by the three machines. When testing this, the technician also measured the diameter of each fiber, and this measurement may be used as a concomitant variable to improve precision of the estimates.

Source

Montgomery, D. C. (2013) *Design and Analysis of Experiments* (8th ed.). John Wiley and Sons, ISBN 978-1-118-14692-7.

Examples

```
require(lsmeans)
fiber.lm <- lm(strength ~ diameter + machine, data=fiber)
ref.grid(fiber.lm)

# Covariate-adjusted means and comparisons
lsmeans(fiber.lm, pairwise ~ machine)
```

---

glht **lsmeans** *support for* glht

Description

These functions and methods provide an interface between **lsmeans** and the [glht](#) function for simultaneous inference in the **multcomp** package.

Usage

```
## S3 method for class 'ref.grid'
as.glht(object, ...)
## S3 method for class 'lsm.list'
as.glht(object, ..., which = 1)

## S3 method for class 'glht.list'
summary(object, ...)

lsm(...)
```

Arguments

object	An object of the required class.
which	Numeric index of which element of the lsm.list to use.
...	Additional arguments to other methods.

Details

lsm is meant to be called only *from* "glht" as its second (linfct) argument. It works similarly to mcp except with specs (and optionally by and contr arguments) provided as in a call to lsmeans.

When there is a non-NULL by variable (either explicitly or implicitly), each "by" group is passed separately to glht and returned as a list of "glht" objects. For convenience, this is classed as "glht.list" and a summary method is provided.

Value

as.glht returns an object of class glht, or of class glht.list if by is non-NULL. The latter is simply a list of glht objects but it has its own summary method which returns a list of summary.glht objects.

Note

There is also a glht method for class ref.grid, but it is far preferable to use as.glht instead, as its model argument is redundant.

Author(s)

Russell V. Lenth

See Also

lsmeans, glht

Examples

```
require(lsmeans)
require(multcomp)

warp.lm <- lm(breaks ~ wool*tension, data = warpbreaks)

# Using 'lsm'
summary(glht(warp.lm, lsm(pairwise ~ tension | wool)))

# Same, but using an existing 'lsmeans' result
warp.lsmobj <- lsmeans(warp.lm, ~ tension | wool)
summary(as.glht(pairs(warp.lsmobj)))

# Same contrasts, but treat as one family
summary(as.glht(pairs(warp.lsmobj), by = NULL))
```

---

lsmeans	*Least-squares means*

Description

Compute least-squares means for specified factors or factor combinations in a linear model, and optionally comparisons or contrasts among them.

Usage

```
## S3 method for class 'character'
lsmeans(object, specs, ...)
## (used when 'specs' is 'character')

## S3 method for class 'character.ref.grid'
lsmeans(object, specs, by = NULL,
    fac.reduce = function(coefs) apply(coefs, 2, mean), contr,
    options = getOption("lsmeans")$lsmeans, weights, ...)
## (used when 'object' is a 'ref.grid' and 'specs' is 'character')

## S3 method for class 'list'
lsmeans(object, specs, ...)
## (used when 'specs' is a 'list')

## S3 method for class 'formula'
lsmeans(object, specs, contr.list, trend, ...)
## (used when 'specs' is a 'formula')

lstrends(model, specs, var, delta.var = 0.01 * rng, data, ...)

lsmobj(bhat, V, levels, linfct, df = NA, post.beta = matrix(NA), ...)
```

Arguments

object	An object of class `ref.grid`; or a fitted model object that is supported, such as the result of a call to `lm` or `lmer`. Many fitted-model objects are supported; see `link{models}` for details.
specs	A `character` vector specifying the names of the predictors over which LS-means are desired. `specs` may also be a `formula` or a `list` (optionally named) of valid specs. Use of formulas is described in the Details section below.
by	A character vector specifying the names of predictors to condition on.
fac.reduce	A function that combines the rows of a matrix into a single vector. This implements the "marginal averaging" aspect of least-squares means. The default is the mean of the rows. Typically if it is overridden, it would be some kind of weighted mean of the rows. If `fac.reduce` is nonlinear, bizarre results are likely, and LS means will not be interpretable. If the `weights` argument is non-missing, `fac.reduce` is ignored.

contr	A `list` of contrast coefficients to apply to the least-squares means – or the root name of an `.lsmc` function that returns such coefficients. In addition, `contr = "cld"` is an alternative way to invoke the [cld](#) function. See [contrast](#) for more details on contrasts. NOTE: `contr` is ignored when `specs` is a formula.
contr.list	A named `list` of `lists` of contrast coefficients, as for `contr`. This is used only in the formula method; see Details below.
options	If non-NULL, a named `list` of arguments to pass to [update](#), just after the object is constructed.
weights	Numeric vector, numeric matrix, or character string specifying weights to use in averaging predictions. If a vector, its length must equal the number of predictions to be averaged to obtain each least-squares mean. If a matrix, each row of the matrix is used in turn, wrapping back to the first row as needed. When in doubt about what is being averaged (or how many), first call with `weights = "show.levels"`.)

If a string, it should partially match one of the following:

"equal" Use an equally weighted average.

"proportional" Weight in proportion to the frequencies (in the original data) of the factor combinations that are averaged over.

"outer" Weight in proportion to each individual factor's marginal frequencies. Thus, the weights for a combination of factors are the outer product of the one-factor margins

"cells" Weight according to the frequencies of the cells being averaged.

show.levels This is a convenience feature for understanding what is being averaged over. Instead of a table of LS means, this causes the function to return a table showing the levels that are averaged over, in the order they appear.

Outer weights are like the 'expected' counts in a chi-square test of independence, and will yield the same results as those obtained by proportional averaging with one factor at a time. All except `"cells"` uses the same set of weights for each mean. In a model where the predicted values are the cell means, cell weights will yield the raw averages of the data for the factors involved. Note: If weights were used in fitting the model, then weight totals are used in place of frequencies in these schemes.

If `weights` is used, `fac.reduce` is ignored.

trend	Including this argument is an alternative way of calling `lstrends` with it as its `var` argument.
model	A supported model object.
var	Character giving the name of a variable with respect to which a difference quotient of the linear predictors is computed. In order for this to be useful, `var` should be a numeric predictor that interacts with at least one factor in `specs`. Then instead of computing least-squares means, we compute and compare the slopes of the `var` trend over levels of the specified other predictor(s). As in least-squares means, marginal averages are computed when some variables in the reference grid are excluded for the specification.

The user may specify some monotone function of one variable, e.g., `var = "log(dose)"`. If so, the chain rule is applied. Note that, in this example, if `model` contains

	log(dose) as a predictor, we will be comparing the slopes estimated by that model, whereas specifying var = "dose" would perform a transformation of those slopes.
delta.var	The value of *h* to use in forming the difference quotient *(f(x+h) - f(x))/h*. Changing it (especially changing its sign) may be necessary to avoid numerical problems such as logs of negative numbers. The default value is 1/100 of the range of var over the dataset.
data	As in [ref.grid](), you may use this argument to supply the dataset used in fitting the model, for situations where it is not possible to reconstruct the data. Otherwise, leave it missing.
bhat	Numeric. Vector of regression coefficients.
V	Square matrix. Covariance matrix of bhat
levels	Named list or vector. Levels of factor(s) that define the estimates defined by linfct. If not a list, we assume one factor named "level"
linfct	Matrix. Linear functions of bhat for each combination of levels
df	Numeric or function with arguments x,dfargs). If a number, that is used for the degrees of freedom. If a function, it should return the degrees of freedom for sum(x*bhat); if additional parameters are needed, include them in ... as dfargs (not abbreviated).
post.beta	Matrix whose columns comprise a sample from the posterior distribution of the regression coefficients (so that typically, the column averages will be bhat). A 1 x 1 matrix of NA indicates that such a sample is unavailable.
...	Additional arguments passed to other methods or to [ref.grid](). For example, vcov. may be used to override the default covariance estimate, and some models allow additional options. Some models require data to be given explicitly. See the help pages for [ref.grid]() and [models](). In addition, if the model formula contains references to variables that are not predictors, you must provide a params argument with a list of their names; see the example below for Oatsq.lm.

Details

Least-squares means are predictions from a linear model over a *reference grid*, or marginal averages thereof. They have been popularized by **SAS** (SAS Institute, 2012). The [ref.grid]() function identifies/creates the reference grid upon which lsmeans is based.

If specs is a formula, it should be of the form contr ~ specs | by. The formula is parsed and then used as the arguments contr, specs, and by as indicated. The left-hand side is optional, but if specified it should be the name of a contrast family (e.g., pairwise) or of a sub-list of contr.list. Operators like * or : are necessary to delineate names in the formulas, but otherwise are ignored.

A number of standard contrast families are provided. They can be identified as functions having names ending in .lsmc – use

ls("package:lsmeans", pat=".lsmc")

to list them. See the documentation for [pairwise.lsmc]() and its siblings for details. You may write your own .lsmc function for custom contrasts.

The function lsmobj may be used to construct an object just like one returned by lsmeans from user-specified coefficients, covariance matrix, levels (or row labels), linear functions for each row, and degrees of freedom. After the object is constructed, it is updateed with any additional arguments in ....

Value

When specs is a character vector or one-sided formula, an object of class lsmobj. A number of methods are provided for further analysis, including summary, confint, test, contrast, pairs, and cld.

When specs is a list or a formula having a left-hand side, the eturn value is an lsm.list object, which is simply a list of lsmobj objects. Methods for lsm.list objects are the same as those for lsmobj, but they apply to only one member of the list, determined by its which argument.

Note

If the model formula contains variables that are not predictors (e.g., degree of a polynomial, knots for a spline, etc.), you must add a params argument to the call

Note

While using specs as a two-sided formula or a list is a convenient way to get a lot of results with minimal effort, it can also create confusion when additional arguments are provided, because not all arguments may be applied to all the results produced (see examples). Thus, the safer route is to do things incrementally.

Note

lsmeans and its relatives can produce fatal errors or incorrect results with models containing splines (e.g., ns) and other smoothers because the required information to reconstruct their basis is not always available. A model with poly involving two or more predictors will almost always produce misleading results without any warning; but poly(..., raw = TRUE) will work correctly.

Note

For a ref.grid or lsmobj object created in **lsmeans** version 2.10 or earlier, the information needed by the weights argument is not present; so a message is displayed and averaging is done using fac.reduce.

Author(s)

Russell V. Lenth

References

SAS Institute Inc. (2012) Online documentation; Shared concepts; LSMEANS statement, http://support.sas.com/documentation/cdl/en/statug/63962/HTML/default/viewer.htm#statug_introcom_a0000003362.htm, accessed August 15, 2012.

See Also

models, pairwise.lsmc, glht

Examples

```
require(lsmeans)

### Covariance example (from Montgomery Design (8th ed.), p.656)
# Uses supplied dataset 'fiber'
fiber.lm <- lm(strength ~ diameter + machine, data = fiber)

# adjusted means and comparisons, treating machine C as control
( fiber.lsm <- lsmeans (fiber.lm, "machine") )
contrast(fiber.lsm, "trt.vs.ctrlk")
# Or get both at once using
#      lsmeans (fiber.lm, "machine", contr = "trt.vs.ctrlk")


### Factorial experiment
warp.lm <- lm(breaks ~ wool * tension, data = warpbreaks)
( warp.lsm <- lsmeans (warp.lm,  ~ wool | tension,
    options = list(estName = "pred.breaks")) )
pairs(warp.lsm) # remembers 'by' structure
contrast(warp.lsm, method = "poly", by = "wool")


### Unbalanced split-plot example ###
#-- The imbalance is imposed deliberately to illustrate that
#-- the variance estimates become biased
require(nlme)
Oats.lme <- lme(yield ~ factor(nitro) + Variety,
    random = ~1 | Block/Variety,
    subset = -c(1,2,3,5,8,13,21,34,55), data = Oats)
(Oats.anal <- lsmeans(Oats.lme, list(poly ~ nitro, pairwise ~ Variety)))


### Issues with lists of specs
test(Oats.anal)                 # Uses 1st element by default
confint(Oats.anal, which = 4)  # or confint(Oats.anal[[4]])

# These two calls apply the "bonf" adjustment to different elements...
lsmeans(warp.lm, eff ~ wool, adjust = "bonf")
lsmeans(warp.lm, eff ~ wool, options = list(adjust = "bonf"))
# Advice: Do one thing at a time if you want non-default adjustments
#   e.g., wool.lsm <- lsmeans(warp.lm, ~ wool)
#          summary(wool.lsm, adjust = "bonf")
#          contrast(wool.lsm, "eff", adjust = "sidak")


### Weights
# See what's being averaged over in the above
lsmeans(Oats.lme, ~ nitro, cov.reduce = FALSE, weights = "show.levels")
```

```
# Give three times the weight to Marvellous
lsmeans(Oats.lme, ~ nitro, cov.reduce = FALSE, weights = c(1,3,1))


### Model with a quadratic trend for 'nitro'
# Also illustrates use of `params` argument to list non-predictors
deg = 2
Oatsq.lm <- lm(yield ~ Block + poly(nitro, deg) + Variety, data = Oats)
# Predictions at each unique 'nitro' value in the dataset
lsmeans(Oatsq.lm, ~ nitro, cov.reduce = FALSE, params = "deg")


### Trends
fiber.lm <- lm(strength ~ diameter*machine, data=fiber)
# Obtain slopes for each machine ...
( fiber.lst <- lstrends(fiber.lm, "machine", var="diameter") )
# ... and pairwise comparisons thereof
pairs(fiber.lst)

# Suppose we want trends relative to sqrt(diameter)...
lstrends(fiber.lm, ~ machine | diameter, var = "sqrt(diameter)",
    at = list(diameter = c(20,30)))

# Given summary statistics for 4 cities computed elsewhere,
# obtain multiple comparisons of their means using the
# Satterthwaite method
ybar <- c(47.6, 53.2, 88.9, 69.8)
s <-    c(12.1, 19.5, 22.8, 13.2)
n <-    c(44,   11,   37,   24)
se2 = s^2 / n
Satt.df <- function(x, dfargs)
    sum(x * dfargs$v)^2 / sum((x * dfargs$v)^2 / (dfargs$n - 1))
city.lsm <- lsmobj(bhat = ybar, V = diag(se2),
    levels = list(city = LETTERS[1:4]), linfct = diag(c(1,1,1,1)),
    df = Satt.df, dfargs = list(v = se2, n = n), estName = "mean")
city.lsm
contrast(city.lsm, "revpairwise")


# See also many other examples in documentation for
# 'contrast', 'cld', 'glht', 'lsmip', 'ref.grid', 'MOats',
# 'nutrition', etc., and in the vignettes
```

---

lsmip   *Least-squares means interaction plot*

Description

This function creates an interaction plot of the least-squares means based on a fitted model and a
simple formula specification.

Usage

```
## Default S3 method:
lsmip(object, formula, type,
    pch = c(1,2,6,7,9,10,15:20),
    lty = 1, col = NULL, ...)
```

Arguments

object	An object of class lsmobj, or a fitted model of a class supported by lsmeans.
formula	Formula of the form trace.factors ~ x.factors	by.factors. The least-squares means are plotted against x.factor for each level of trace.factors. by.factors is optional, but if present, it determines separate panels. Each element of this formula may be a single factor in the model, or a combination of factors using the * operator.
type	As in predict, this determines whether we want to inverse-transform the predictions ('type="response"') or not (any other choice). The default is "link", unless the "predict.type" option is in force; see lsm.options.
pch	The plotting characters to use for each group (i.e., levels of trace.factors). They are recycled as needed.
lty	The line types to use for each group. Recycled as needed.
col	The colors to use for each group, recycled as needed. If not specified, the default trellis colors are used.
...	Additional arguments passed to lsmeans or to xyplot.

Details

If object is a fitted model, lsmeans is called with an appropriate specification to obtain least-squares means for each combination of the factors present in formula (in addition, any arguments in ... that match at, trend, cov.reduce, or fac.reduce are passed to lsmeans). Otherwise, if object is an lsmobj object, its first element is used, and it must contain one lsmean value for each combination of the factors present in formula.

Value

(Invisibly), the table of least-squares means that were plotted.

Note

This function uses the xyplot function in the lattice package (an error is returned if lattice is not installed). Conceptually, it is equivalent to interaction.plot where the summarization function is the least-squares means.

Author(s)

Russell V. Lenth

See Also

interaction.plot

Examples

```
require(lsmeans)
require(lattice)

#--- Two-factor example
warp.lm <- lm(breaks ~ wool * tension, data = warpbreaks)

# Following plot is the same as the usual interaction plot of the data
lsmip(warp.lm, wool ~ tension)

#--- Three-factor example
noise.lm = lm(noise ~ size * type * side, data = auto.noise)

# Separate interaction plots of size by type, for each side
lsmip(noise.lm, type ~ size | side)

# One interaction plot, using combinations of size and side as the x factor
lsmip(noise.lm, type ~ side * size)

# One interaction plot using combinations of type and side as the trace factor
# customize the colors, line types, and symbols to suggest these combinations
lsmip(noise.lm, type * side ~ size, lty=1:2, col=1:2, pch=c(1,1,2,2))

# 3-way interaction is significant, but doesn't make a lot of visual difference...
noise.lm2 = update(noise.lm, . ~ . - size:type:side)
lsmip(noise.lm2, type * side ~ size, lty=1:2, col=1:2, pch=c(1,1,2,2))
```

MOats   *Oats data in multivariate form*

Description

This is the `Oats` dataset provided in the **nlme** package, but it is rearranged as one multivariate observation per plot.

Usage

```
data(MOats)
```

Format

A data frame with 18 observations on the following 3 variables.

`Variety` a factor with levels `Golden Rain`, `Marvellous`, `Victory`

`Block` an ordered factor with levels VI < V < III < IV < II < I

`yield` a matrix with 4 columns, giving the yields with nitrogen concentrations of 0, .2, .4, and .6.

Details

These data arise from a split-plot experiment reported by Yates (1935) and used as an example in Pinheiro and Bates (2000) and other texts. Six blocks were divided into three whole plots, randomly assigned to the three varieties of oats. The whole plots were each divided into 4 split plots and randomized to the four concentrations of nitrogen.

Source

The dataset [Oats](#) in the **nlme** package.

References

Pinheiro, J. C. and Bates D. M. (2000) *Mixed-Effects Models in S and S-PLUS*, Springer, New York. (Appendix A.15)

Yates, F. (1935) Complex experiments, *Journal of the Royal Statistical Society* Suppl. 2, 181-247

Examples

```
require(lsmeans)
MOats.lm <- lm (yield ~ Block + Variety, data = MOats)
MOats.rg <- ref.grid (MOats.lm, mult.name = "nitro")
lsmeans(MOats.rg, ~ nitro | Variety)
```

---

models   *Models supported in* **lsmeans**

Description

Here we document what model objects may be used with **lsmeans**, and some special features of some of them. We start with those in the **stats** package; the other packages follow in alphabetical order.

Certain objects are affected by optional arguments to functions that construct ref.grid or lsmobj objects, including [ref.grid](#), [lsmeans](#), [lstrends](#), and [lsmip](#). When "arguments" are mentioned in the subsequent object-by-object documentation, we are talking about arguments in these constructors.

Additional models can be supported by writing appropriate recover.data and lsm.basis methods. See [extending-lsmeans](#) and vignette("extending") for details.

stats package

lm, aov, glm No extended features. Note that the lm support often extends to a number of model objects that inherit from it, such as rlm in the **MASS** package and rsm in the **rsm** package.

mlm, maov, manova When there is a multivariate response, the different responses are treated as if they were levels of a factor – named rep.meas by default. The mult.name argument may be used to change this name. The mult.levs argument may specify a named list of one or more sets of levels. If this has more than one element, then the multivariate levels are expressed as combinations of the named factor levels via the function [expand.grid](#).

aovlist Support for these objects is limited. To avoid strong biases in the predictions, the `contrasts` attribute of all factors should be of a type that sums to zero – for example, `"contr.sum"`, `"contr.poly"`, or `"contr.helmert"` but *not* `"contr.treatment"`. Only intra-block estimates of covariances are used. That is, if a factor appears in more than one error stratum, only the covariance structure from its lowest stratum is used in estimating standard errors. Degrees of freedom are obtained using the Satterthwaite method. In general, aovList support is best with balanced designs, and due caution in the use of contrasts. If a `vcov.` argument is supplied, it must yield a single covariance matrix for the unique fixed effects, and the degrees of freedom are set to `NA`.

afex package

mixed Support for the `full.model` element of these objects is the same as that for `merMod` in the **lme4** package – see below. However, for **afex** versions 0.10-113 and earlier, the `data` argument is required in calls to `lsmeans` or `ref.grid`, as the information about the original dataset is not preserved in the object.

CARBayes package

carbayes The user *must* supply (via the `data` argument) the dataset used in fitting the model. As with other MCMC-based objects, the summaries and such are frequentist, but the `as.mcmc` method provides a posterior sample of the desired quantities.

coxme package

coxme No additional options. Support for these models is experimental; may throw errors or incorrect results.

gam package

gam Currently, gam objects are not supported. Past versions of **lsmeans** appeared to support gam models owing to inheritance from `lm`, but the results were incorrect because spline features were ignored. We now explicitly trap gam objects to avoid tyhese misleading analyses.

gee and geepack packages

These models all have more than one covariance estimate available, and it may be selected by supplying a string as the `vcov.method` argument. It is partially matched with the available choices; thus, for example, '`vcov = "n"`' translates to '`vcov.method = "naive"`'

gee Available covariance estimates are specified in `vcov.method` as `"robust"` (the default) and `"naive"`.

geeglm, geese Available covariance estimates are specified in `vcov.method` as `"vbeta"` (the default), `"vbeta.naiv"`, `"vbeta.j1s"`, or `"vbeta.fij"`. The aliases `"robust"` (for `"vbeta"`) and `"naive"` (for `"vbeta.naiv"` are also accepted.

If a matrix or function is supplied as `vcov.method`, it is interpreted as a `vcov.` specification as described for `...` in `ref.grid`.

glmmADMB package

> **glmmadmb** No extended features.

lme4 package

> **lmerMod** If the **pbkrtest** package is installed, degrees of freedom for confidence intervals and tests are obtained using its `ddf_Lb` function, and the covariance matrix is adjusted using `vcovAdj`. If **pbkrtest** is not installed, the covariance matrix is not adjusted, degrees of freedom are set to NA, and asymptotic results are displayed.
>
> The user may disable the use of **pbkrtest** via 'lsm.options(disable.pbkrtest=TRUE)' (this does not disable the **pbkrtest** package entirely, just its use in **lsmeans**). The df argument may be used to specify some other degrees of freedom. Specifying df is *not* equivalent to disabling **pbkrtest**, because if not disabled, the covariance matrix is still adjusted.
>
> **glmerMod** No degrees of freedom are available for these objects, so tests and confidence intervals are asymptotic.

lme4.0 package

> **mer** Only asymptotic results are available (no d.f.).

MASS package

> **glmmPQL** Supported by virtue of inheritance from `lme` in the **nlme** package.
>
> **glm.nb** Supported by virtue of inheritance from `glm`.
>
> **polr** There are two optional arguments: `mode` and `rescale` (which defaults to 'c(0,1)'). For details, see the documentation below regarding the support for the **ordinal** package, which produces comparable objects (but since `polr` does not support scale models, `mode="scale"` is not supported). Tests and confidence intervals are asymptotic.
>
> **rlm** Supported by virtue of inheritance from `lm`.

MCMCglmm package

> **MCMCglmm** Currently, I have found no way to reconstruct the data based on information in the object; thus, you *must* provide the dataset via the `data` argument. In addition, the `contrasts` specifications are not recoverable from the object, so the system default must match what was actually used in fitting the model. The usual `summary`, `test`, etc. methods provide frequentist analyses of the results based on the posterior means and covariances. However, an `as.mcmc` method is provided that creates an `mcmc` object that can be summarized or plotted using the **coda** package. It provides a posterior sample of lsmeans for the given reference grid, based on the posterior sample of the fixed effects from the MCMCglmm object.

MCMCpack package (and perhaps others)

> **mcmc** Certain linear-model or mixed-model objects are of class `mcmc`, and contain a sample from the posterior distribution of fixed-effect coefficients. In some cases (e.g., results of MCMCregress and MCMCpoisson), the object may include a "call" attribute that lsmeans can use to reconstruct the data and obtain a basis for the least-squares means. If not, a formula and data argument are provided that may help produce the right results. In addition, the contrasts

specifications are not recoverable from the object, so the system default must match what was actually used in fitting the model. As for other MCMC-based objects, the summaries and such are frequentist, but the as.mcmc method provides a posterior sample of the desired quantities.

nlme package

gls No additional features. Degrees of freedom are computed using N - p in object$dims. This is consistent with nlme:::summary.gls but seems questionable.

lme Degrees of freedom are obtained using a containment method, i.e., the minimum of those elements of object$fixDF$X receiving nonzero weight (but with a correction to the lme object's intercept df). (This is similar to **SAS**'s containment method, but I believe **SAS** does it incorrectly when the estimands are not contrasts.) The optional argument adjustSigma (defaults to TRUE) will adjust standard errors like in [summary.lme](#) when the model is fitted using the "ML" method. **Note:** It is possible for the adjust argument (for p-value adjustments) to conflict with adjustSigma. The workaround is to specify both: e.g., 'lsmeans(mod.lme, pairwise ~ trt, adjustSigma = TRUE,

nlme Support is provided for inferences on parameters named in the fixed part of the model. The user *must* specify param in the call and give the name of a parameter that appears in the right-hand side of a fixed formula. Degrees of freedom are obtained using the containment-like method described above for lme.

ordinal package

clm,clmm The reference grid will include all variables that appear in the main model as well as those in the scale or nominal models. There are two optional arguments: mode (a character string) and rescale (which defaults to 'c(0,1)'). mode should match one of "latent" (the default), "linear.predictor", "cum.prob", "exc.prob", "prob", "mean.class", or "scale".

With 'mode = "latent"', the reference-grid predictions are made on the scale of the latent variable implied by the model. The scale and location of this latent variable are arbitrary, and may be altered via rescale. The predictions are multiplied by 'rescale[2]', then 'rescale[1]' is added. Keep in mind that the scaling is related to the link function used in the model; for example, changing from a probit link to a logistic link will inflate the latent values by around $\pi/\sqrt{3}$, all other things being equal. rescale has no effect for other values of mode.

With 'mode = "linear.predictor"' mode = "cum.prob", and mode = "exc.prob", the boundaries between categories (i.e., thresholds) in the ordinal response are included in the reference grid as a pseudo-factor named cut. The reference-grid predictions are then of the cumulative probabilities at each threshold (for mode = "cum.prob"), exceedance probabilities (one minus cumulative probabilities, for mode = "exc.prob"), or the link function thereof (for mode = "linear.predictor").

With mode = "prob", a pseudo-factor with the same name as the model's response variable is created, and the grid predictions are of the probabilities of each class of the ordinal response. With "mean.class", the returned results are means of the ordinal response, interpreted as a numeric value from 1 to the number of classes, using the "prob" results as the estimated probability distribution for each case.

With mode = "scale", and the fitted object incorporates a scale model, least-squares means are obtained for the factors in the scale model instead of the response model. The grid is constructed using only the factors in the scale model.

Any grid point that is non-estimable by either the location or the scale model (if present) is set to NA, and any LS-means involving such a grid point will also be non-estimable. A consequence of this is that if there is a rank-deficient scale model, and then *all* latent responses become non-estimable because the predictions are made using the average log-scale estimate.

Tests and confidence intervals are asymptotic.

rms package

Potential masking issue Both **rms** and **lsmeans** offer contrast methods, and whichever package is loaded later masks the other. Thus, you may need to call lsmeans::contrast or rms::contrast explicitly to access the one you want.

Objects inheriting from rms Standard support is provided. However, with models having more than one intercept (e.g. from orm), a mode argument is provided that works similarly to that for the **ordinal** package. The available modes are "middle" (the default), "latent", "linear.predictor", "cum.prob", "exc.prob", "prob", and "mean.class". All are as described for the **ordinal** package, except as noted below.

With mode = "middle" (this is the default), the middle intercept is used, comparable to the default for **rms**'s Predict function. This is quite similar in concept to mode = "latent", where all intercepts are averaged together.

Results for mode = "linear.predictor" are reversed from those in the **ordinal** package, because orm models predict the link function of the *upper*-tail (exceedance) probabilities.

With mode = "prob", a pseudo-factor is created having the same name as the model response variable, but its levels are always integers '1, 2, ...' regardless of the levels of the original response.

survival package

survreg, coxph No extended features.

Author(s)

Russell V. Lenth

See Also

[ref.grid](), [lsm.basis]()

---

nutrition	*Nutrition data*

Description

This observational dataset involves three factors, but where several factor combinations are missing. It is used as a case study in Milliken and Johnson, Chapter 17, p.202. (You may also find it in the second edition, p.278.)

Usage

```
nutrition
```

Format

A data frame with 107 observations on the following 4 variables.

age a factor with levels 1, 2, 3, 4. Mother's age group.

group a factor with levels FoodStamps, NoAid. Whether or not the family receives food stamp assistance.

race a factor with levels Black, Hispanic, White. Mother's race.

gain a numeric vector (the response variable). Gain score (posttest minus pretest) on knowledge of nutrition.

Details

A survey was conducted by home economists "to study how much lower-socioeconomic-level mothers knew about nutrition and to judge the effect of a training program designed to increase therir knowledge of nutrition." This is a messy dataset with several empty cells.

Source

Milliken, G. A. and Johnson, D. E. (1984) *Analysis of Messy Data – Volume I: Designed Experiments*. Van Nostrand, ISBN 0-534-02713-7.

Examples

```
require(lsmeans)
nutr.aov <- aov(gain ~ (group + age + race)^2, data = nutrition)

# Summarize predictions for age group 3
nutr.lsm <- lsmeans(nutr.aov, ~ race * group,
                at = list(age="3"))

lsmip(nutr.lsm, race ~ group)

# Hispanics seem exceptional; but, this doesn't test out due to very sparse data
cld(nutr.lsm, by = "group")
cld(nutr.lsm, by = "race")
```

---

oranges   *Orange sales*

Description

This example dataset on sales of oranges has two factors, two covariates, and two responses. There is one observation per factor combination.

Usage

```
data(oranges)
```

Format

A data frame with 36 observations on the following 6 variables.

store a factor with levels 1 2 3 4 5 6. The store that was observed.

day a factor with levels 1 2 3 4 5 6. The day the observation was taken (same for each store).

price1 a numeric vector. Price of variety 1.

price2 a numeric vector. Price of variety 2.

sales1 a numeric vector. Sales (per customer) of variety 1.

sales2 a numeric vector. Sales (per customer) of variety 2.

Source

Download from <http://ftp.sas.com/samples/A56655>.

References

Littell, R., Stroup W., Freund, R. (2002) *SAS For Linear Models* (4th edition). SAS Institute. ISBN 1-59047-023-0.

Examples

```
require(lsmeans)

# Example on p.244 of Littell et al.
oranges.lm <- lm(sales1 ~ price1*day, data = oranges)
lsmeans(oranges.lm, "day")

# Example on p.246
lsmeans(oranges.lm, "day", at = list(price1 = 0))
```

---

pairwise.lsmc   *Contrast families*

Description

These functions return standard sets of contrast coefficients. The name of any of these functions (with the .lsmc omitted) may be used as the method argument in [contrast](#), or as the contr argument or left-hand side of a spec formula in [lsmeans](#).

Usage

```
pairwise.lsmc(levs, ...)
revpairwise.lsmc(levs, ...)
tukey.lsmc(levs, reverse = FALSE)

poly.lsmc(levs, max.degree = min(6, k - 1))

trt.vs.ctrl.lsmc(levs, ref = 1)
trt.vs.ctrl1.lsmc(levs, ...)
trt.vs.ctrlk.lsmc(levs, ...)
dunnett.lsmc(levs, ref = 1)

consec.lsmc(levs, reverse = FALSE, ...)
mean_chg.lsmc(levs, reverse = FALSE, ...)

eff.lsmc(levs, ...)
del.eff.lsmc(levs, ...)
```

Arguments

`levs`	Vector of factor levels
`...`	Additional arguments, ignored but needed to make these functions interchangeable
`max.degree`	The maximum degree of the polynomial contrasts in `poly.lsmc`
`reverse`	Logical value to determine the direction of comparisons, e.g., pairwise (if `TRUE`) or reverse-pairwise (if `FALSE`) comparisons.
`ref`	Reference level (or control group) in `trt.vs.ctrl.lsmc`

Details

Each contrast family has a default multiple-testing adjustment as noted below. These adjustments are often only approximate; for a more exacting adjustment, use the interfaces provided to [glht](#) in the **multcomp** package.

`pairwise.lsmc`, `revpairwise.lsmc`, and `tukey.lsmc` generate contrasts for all pairwise comparisons among least-squares means at the levels in `levs`. The distinction is in which direction they are subtracted. For factor levels A, B, C, D, `pairwise.lsmc` generates the comparisons A-B, A-C, A-D, B-C, B-D, and C-D, whereas `revpairwise.lsmc` generates B-A, C-A, C-B, D-A, D-B, and D-C. `tukey.lsmc` invokes `pairwise.lsmc` or `revpairwise.lsmc` depending on `reverse`. The default multiplicity adjustment method is `"tukey"`, which is approximate when the standard errors differ.

`poly.lsmc` generates orthogonal polynomial contrasts, assuming equally-spaced factor levels. These are derived from the [poly](#) function, but an ad hoc algorithm is used to scale them to integer coefficients that are (usually) the same as in published tables of orthogonal polynomial contrasts. The default multiplicity adjustment method is `"none"`.

`trt.vs.ctrl.lsmc` and its relatives generate contrasts for comparing one level (or the average over specified levels) with each of the other levels. The argument `ref` should be the *index*(es)

(not the labels) of the reference level(s). `trt.vs.ctrl1.lsmc` is the same as `trt.vs.ctrl` with a reference value of 1, and `trt.vs.ctrlk.lsmc` is the same as `trt.vs.ctrl` with a reference value of `length(levs)`. `dunnett.lsmc` is the same as `trt.vs.ctrl`. The default multiplicity adjustment method is `"dunnettx"`, a close approximation to the Dunnett adjustment.

`consec.lsmc` and `mean_chg.lsmc` are useful for contrasting treatments that occur in sequence. For a factor with levels A, B, C, D, E, `consec.lsmc` generates the comparisons B-A, C-B, and D-C, while `mean_chg.lsmc` generates the contrasts (B+C+D)/3 - A, (C+D)/2 - (A+B)/2, and D - (A+B+C)/3. With `reverse = TRUE`, these differences go in the opposite direction.

`eff.lsmc` and `del.eff.lsmc` generate contrasts that compare each level with the average over all levels (in `eff.lsmc`) or over all other levels (in `del.eff.lsmc`). These differ only in how they are scaled. For a set of k lsmeans, `del.eff.lsmc` gives weight 1 to one lsmean and weight $-1/(k-1)$ to the others, while `eff.lsmc` gives weights $(k-1)/k$ and $-1/k$ respectively, as in subtracting the overall lsmean from each lsmean. The default multiplicity adjustment method is `"fdr"`. This is a Bonferroni-based method and is slightly conservative; see `p.adjust`

Value

A `data.frame`, each column containing contrast coefficients for `levs`. The `"desc"` attribute is used to label the results in `lsmeans`, and the `"adjust"` attribute gives the default adjustment method for multiplicity.

Note

You may create your own contrast functions, using these as guides. A function named `mycontr.lsmc` may be invoked in `lsmeans` via, e.g.,

`lsmeans(\var{object}, mycontr ~ \var{factor})`

The `"desc"`, `"adjust"`, and `"offset"` attributes are optional; if present, these are passed to `contrast`. If absent, the root name of the function is used as `"desc"`, and no adjustment is requested for p values. See the examples.

Author(s)

Russell V. Lenth

See Also

`lsmeans`, `glht`

Examples

```
### View orthogonal polynomials for 4 levels
poly.lsmc(1:4)

### Setting up a custom contrast function
helmert.lsmc <- function(levs, ...) {
  M <- as.data.frame(contr.helmert(levs))
  names(M) <- paste(levs[-1],"vs earlier")
  attr(M, "desc") <- "Helmert contrasts"
```

```
    M
}
warp.lm <- lm(breaks ~ wool*tension, data = warpbreaks)
lsmeans(warp.lm, helmert ~ tension | wool)
```

recover.data   *Support functions for creating a reference grid*

Description

This documents the methods used to create a ref.grid object from a fitted model.

Usage

```
recover.data(object, ...)
## S3 method for class 'call'
recover.data(object, trms, na.action,
    data = NULL, params = NULL, ...)

lsm.basis(object, trms, xlev, grid, ...)

All.vars(expr, retain = c("\$", "\\[\\[", "\\]\\]"), ...)
```

Arguments

object	An object returned from a model-fitting function.
trms	The terms component of object
xlev	Named list of levels of factors in the model frame. This should *not* include levels of factors created in the model itself, e.g., by including a factor call in the model formula.
grid	A data.frame containing predictor values at which predictions are needed.
na.action	Integer vector of indices of observations to ignore; or NULL if none
data	Data frame. Usually, this is NULL. However, if non-null, this is used in place of the reconstructed dataset. It must have all of the predictors used in the model, and any factor levels must match those used in fitting the model.
params	Character vector giving the names of any variables in the model formula that are *not* predictors. An example would be a variable knots specifying the knots to use in a spline model.
expr	A formula
retain	Character vector of operators to retain (escaped as for gsub)
...	Additional arguments passed to other methods.

Details

To create a reference grid, the `ref.grid` function needs to reconstruct the data used in fitting the model, and then obtain a matrix of linear functions of the regression coefficients for a given grid of predictor values. These tasks are performed by calls to `recover.data` and `lsm.basis` respectively.

To extend **lsmeans**'s support to additional model types, one need only write S3 methods for these two functions. The existing methods serve as helpful guidance for writing new ones. Most of the work for `recover.data` can be done by its method for class `"call"`, providing the `terms` component and `na.action` data as additional arguments. Writing an `lsm.basis` method is more involved, but the existing methods (e.g., `lsmeans:::lsm.basis.lm`) can serve as models. See the "Value" section below for details on what it needs to return. Also, certain `recover.data` and `lsm.basis` methods are exported from **lsmeans**, so if your object is based on another model-fitting object, it may be that all that is needed is to call one of these exported methods and perhaps make modifications to the results. Contact the developer if you need others of these exported.

If the model has a multivariate response, `bhat` needs to be "flattened" into a single vector, and `X` and `V` must be constructed consistently.

In models where a non-full-rank result is possible (often you can tell by seeing if there is a `singular.ok` argument in the model-fitting function), `summary` and `predict` check the estimability of each prediction, using the [nonest.basis](#) function in the **estimability** package.

The models already supported are detailed in [models](#). Some packages may provide additional **lsmeans** support for its object classes.

Value

`recover.data` should return a `data.frame` containing all the variables in the original data that appear as predictors in the model. Several attributes need to be included as well; see the code for `lsmeans:::recover.data.lm`.

`lsm.basis` should return a `list` with the following elements:

X	The matrix of linear fcns over `grid`, having the same number of rows as `grid` and the number of columns equal to the length of `bhat`.
bhat	The vector of regression coefficients for fixed effects. This should *include* any NAs that result from rank deficiencies.
nbasis	A matrix whose columns form a basis for non-estimable functions of beta, or a 1x1 matrix of NA if there is no rank deficiency.
V	The estimated covariance matrix of `bhat`.
dffun	A function of (k, dfargs) that returns the degrees of freedom associated with sum(k * bhat).
dfargs	A `list` containing additional arguments needed for `dffun`.

`All.vars` is an enhancement of [all.vars](#), whereby the operators specified in `retain` are left intact. Thus, `All.vars(foo$y ~ bar[[2]])` returns `"foo$y"`, `"bar[[2]]"`, whereas `all.vars` returns `"foo"`, `"y"`, `"bar"`

Optional hooks

Some models may need something other than standard linear estimates and standard errors. If so, custom functions may be pointed to via the items misc$estHook, misc$vcovHook and misc$postGridHook. If just the name of the hook function is provided as a character string, then it is retrieved using get.

The estHook function should have arguments '(object, do.se, tol, ...)' where object is the ref.grid or lsmobj object, do.se is a logical flag for whether to return the standard error, and tol is the tolerance for assessing estimability. It should return a matrix with 3 columns: the estimates, standard errors (NA when do.se==FALSE), and degrees of freedom (NA for asymptotic). The number of rows should be the same as 'object@linfct'. The vcovHook function should have arguments '(object, tol, ...)' as described. It should return the covariance matrix for the estimates. Finally, postGridHook, if present, is called at the very end of ref.grid; it takes one argument, the constructed object, and should return a suitably modifiedref.grid object.

Author(s)

Russell V. Lenth

See Also

models, ref.grid, ref.grid-class

Examples

```
## Not run:
    require(lsmeans)

    # Fit a 2-factor model with two empty cells
    warpsing.lm <- lm(breaks ~ wool*tension,
        data = warpbreaks, subset = -(16:40))

    lsmeans:::recover.data.lm(warpsing.lm, data = NULL)
    grid = with(warpbreaks,
        expand.grid(wool = levels(wool), tension = levels(tension)))
    lsmeans:::lsm.basis.lm(warpsing.lm, delete.response(terms(warpsing.lm)),
        warpsing.lm$xlevels, grid)

## End(Not run)

# Compare results of all.vars and All.vars
form = log(cows$weight) ~ factor(bulls[[3]]) * herd$breed
all.vars(form)
All.vars(form)

all.vars(form, functions = TRUE) ## same as all.names(form, unique = TRUE)
All.vars(form, functions = TRUE)
```

---

ref.grid   *Create a reference grid from a fitted model*

Description

Using a fitted model object, determine a reference grid for which least-squares means are defined. The resulting ref.grid object encapsulates all the information needed to calculate LS means and make inferences on them.

Usage

```
ref.grid(object, at, cov.reduce = mean, mult.name, mult.levs,
    options = lsm.options()$ref.grid, data, type, ...)
```

Arguments

object
: An object produced by a supported model-fitting function, such as lm. Many models are supported. See [models](#).

at
: Optional named list of levels for the corresponding variables

cov.reduce
: A function, logical value, or formula; or a named list of these. Each covariate *not specified in* at is reduced according to these specifications.

  If a single function, it is applied to each covariate.

  If logical and TRUE, mean is used. If logical and FALSE, it is equivalent to specifying 'function(x) sort(unique(x))', and these values are considered part of the reference grid; thus, it is a handy alternative to specifying these same values in at.

  If a formula (which must be two-sided), then a model is fitted to that formula using [lm](#); then in the reference grid, its response variable is set to the results of [predict](#) for that model, with the reference grid as newdata. (This is done *after* the reference grid is determined.) A formula is appropriate here when you think experimental conditions affect the covariate as well as the response.

  If cov.reduce is a named list, then the above criteria are used to determine what to do with covariates named in the list. (However, formula elements do not need to be named, as those names are determined from the formulas' left-hand sides.) Any unresolved covariates are reduced using "mean".

  Any cov.reduce specification for a covariate also named in at is ignored.

mult.name
: Character, the name to give to the "factor" whose levels delineate the elements of a multivariate response. If this is provided, it overrides the default name, e.g., "rep.meas" for an [mlm](#) object or "cut" for a [polr](#) object.

mult.levs
: A named list of levels for the dimensions of a multivariate response. If there is more than one element, the combinations of levels are used, in [expand.grid](#) order. The (total) number of levels must match the number of dimensions. If mult.name is specified, this argument is ignored.

options
: If non-NULL, a named list of arguments to pass to [update](#), just after the object is constructed.

data	A data.frame to use to obtain information about the predictors (e.g. factor levels). If missing, then recover.data is used to attempt to reconstruct the data.
type	If provided, this is saved as the "predict.type" setting. See update
...	Optional arguments passed to lsm.basis, such as vcov. (see Details below) or options for certain models (see models).

Details

The reference grid consists of combinations of independent variables over which predictions are made. Least-squares means are defined as these predictions, or marginal averages thereof. The grid is determined by first reconstructing the data used in fitting the model (see recover.data), or by using the data.frame provided in context. The default reference grid is determined by the observed levels of any factors, the ordered unique values of character-valued predictors, and the results of cov.reduce for numeric predictors. These may be overridden using at.

Ability to support a particular class of object depends on the existence of recover.data and lsm.basis methods – see extending-lsmeans for details. The call methods("recover.data") will help identify these.

In certain models, (e.g., results of glmer.nb), it is not possible to identify the original dataset. In such cases, we can work around this by setting data equal to the dataset used in fitting the model, or a suitable subset. Only the complete cases in data are used, so it may be necessary to exclude some unused variables. Using data can also help save computing, especially when the dataset is large. In any case, data must represent all factor levels used in fitting the model. It *cannot* be used as an alternative to at. (Note: If there is a pattern of NAs that caused one or more factor levels to be excluded when fitting the model, then data should also exclude those levels.)

By default, the variance-covariance matrix for the fixed effects is obtained from object, usually via its vcov method. However, the user may override this via a vcov. argument, specifying a matrix or a function. If a matrix, it must be square and of the same dimension and parameter order of the fixed efefcts. If a function, must return a suitable matrix when it is called with object as its only argument.

Value

An S4 object of class "ref.grid" (see ref.grid-class). These objects encapsulate everything needed to do calculations and inferences for least-squares means, and contain nothing that depends on the model-fitting procedure.

Author(s)

Russell V. Lenth

See Also

See also summary and other methods for the returned objects. Reference grids are fundamental to lsmeans. Click here for more on the ref.grid class. Supported models are detailed in models.

Examples

```
require(lsmeans)

fiber.lm <- lm(strength ~ machine*diameter, data = fiber)
ref.grid(fiber.lm)
summary(ref.grid(fiber.lm))

ref.grid(fiber.lm, at = list(diameter = c(15, 25)))

## Not run:
# We could substitute the sandwich estimator vcovHAC(fiber.lm)
# as follows:
require(sandwich)
summary(ref.grid(fiber.lm, vcov. = vcovHAC))

## End(Not run)

# If we thought that the machines affect the diameters
# (admittedly not plausible in this example), then we should use:
ref.grid(fiber.lm, cov.reduce = diameter~machine)

# Multivariate example
MOats.lm = lm(yield ~ Block + Variety, data = MOats)
ref.grid(MOats.lm, mult.name = "nitro")
# silly illustration of how to use 'mult.levs'
ref.grid(MOats.lm, mult.levs = list(T=LETTERS[1:2], U=letters[1:2]))
```

---

ref.grid-class   *Classes* "ref.grid" *and* "lsmobj"

Description

A reference grid encapsulates everything needed to compute least-squares means, independently of the underlying model object. The "lsmobj" class is a minor extension of "ref.grid" where the linear predictors for the reference grid are transformed in some linear way such as marginal averages or contrasts.

Objects from the Classes

Objects of class "ref.grid" are most commonly created by calling the ref.grid function.

Objects of class "lsmobj" are created by calling lsmeans or a related function such as contrast.

Slots

model.info: Object of class "list" containing the elements call (the call that produced the model), terms (its terms object), and xlev (factor-level information)

roles: Object of class "list" containing at least the elements predictors, responses, and multresp. These are character vectors of names of these variables.

grid: Object of class "data.frame" containing the combinations of the variables that define the reference grid. In addition, there is an auxiliary column named ".wgt." holding the observed frequencies or weights for each factor combination (excluding covariates). If the model has one or more offset() calls, there is an another auxiliary column named ".offset.". Auxiliary columns are not considered part of the reference grid. (However, any variables included in offset calls *are* in the reference grid.)

levels: Object of class "list" with each entry containing the distinct levels of variables in the reference grid. Note that grid is obtained by applying the function expand.grid to this list

matlevs: Object of class "list" Like levels but has the levels of any matrices in the original dataset. Matrix columns must always be reduced to a single value for purposes of the reference grid

linfct: Object of class "matrix" giving the linear functions of the regression coefficients for predicting each element of the reference grid. The rows of this matrix go in one-to-one correspondence with the rows of grid, and the columns with elements of bhat

bhat: Object of class "numeric" with the regression coefficients. If there is a multivariate response, this must be flattened to a single vector, and linfct and V redefined appropriately. Important: bhat must *include* any NA values produced by collinearity in the predictors. These are taken care of later in the estimability check.

nbasis: Object of class "matrix" with the basis for the non-estimable functions of the regression coefficients. Every LS mean will correspond to a linear combination of rows of linfct, and that result must be orthogonal to all the columns of nbasis in order to be estimable. This will be NULL if everything is estimable

V: Object of class "matrix", the symmetric variance-covariance matrix of bhat

dffun, dfargs: Objects of class "function" and "list" respectively. dffun(k,dfargs) should return the degrees of freedom for the linear function sum(k*bhat), or NA if unavailable

misc: A list containing additional information used by methods. These include at least the following: estName (the label for the estimates of linear functions), and the default values of infer, level, and adjust to be used in the summary method. Elements in this slot may be modified if desired using the update method.

post.beta: A matrix containing a sample from the posterior distribution of the regression coefficients; or a 1 x 1 matrix of NA if this is not available. When it is non-trivial, the as.mcmc method returns post.beta times t(linfct), which is a sample from the posterior distribution of the LS means.

Extends

Class "lsmobj" extends Class "ref.grid", directly. There is hardly a difference between these classes except for how the slots linfct and grid are obtained, and their show methods.

Methods

All methods for these objects are S3 methods except for show.

show: Prints the results of str for ref.grid objects, and summary for lsmobj objects.

str: Displays a brief listing of the variables and levels defining the grid.

summary: Displays a summary of estimates, standard errors, degrees of freedom, and optionally, tests and/or confidence intervals.

lsmeans: Computes least-squares means and creates an "lsmobj" object.

confint: Confidence intervals for lsmeans.

test: Hypothesis tests.

cld: Compact-letter display for tests of pairwise comparisons

contrast: Contrasts among lsmeans.

pairs: A special case of contrasts for pairwise comparisons.

update: Change defaults used primarily by summary, such as transformation, p-value adjustment, and confidence level.

Author(s)

Russell V. Lenth

See Also

[ref.grid](), [lsmeans]()

Examples

```
showClass("ref.grid")
showClass("lsmobj")
```

---

summary   *Methods for* ref.grid *objects*

Description

Use these methods to summarize, print, plot, or examine objects of class "ref.grid". They also apply to the class "lsmobj", which is an extension of "ref.grid".

Usage

```
## S3 method for class 'ref.grid'
summary(object, infer, level, adjust, by, type, df,
    null = 0, delta = 0, side = 0, ...)

## S3 method for class 'ref.grid'
predict(object, type, ...)

## S3 method for class 'ref.grid'
str(object, ...)

## S3 method for class 'ref.grid'
print(x, ...)
```

```
## S3 method for class 'summary.ref.grid'
print(x, ..., digits = NULL, quote = FALSE, right = TRUE)


## S3 method for class 'lsmobj'
plot(x, y, type, intervals = TRUE, comparisons = FALSE,
    alpha = 0.05, adjust = "tukey", int.adjust = "none", ...)
## S3 method for class 'summary.ref.grid'
plot(x, y, horizontal = TRUE,
    xlab, ylab, layout, ...)


## S3 method for class 'ref.grid'
vcov(object, ...)


regrid (object, transform = TRUE)


## S3 method for class 'ref.grid'
as.mcmc(x, names = TRUE, ...)
```

Arguments

object	An object of class "ref.grid".
infer	A vector of two logical values. The first determines whether confidence intervals are displayed, and the second determines whether t tests and P values are displayed. If only one value is provided, it is used for both.
level	Confidence level for confidence intervals, if infer[1] is TRUE.
adjust	Character value naming the method used to adjust p values or confidence limits; or to adjust comparison arrows in plot. See Details.
by	Character name(s) of variables to use for grouping. This affects the family of tests considered in adjusted P values. The printed display of the summary is grouped by the by variables.
type	Type of prediction desired. This only has an effect if there is a known transformation or link function. "response" specifies that the inverse transformation be applied. Other valid values are "link", "lp", and "linear"; these are equivalent, and request that results be shown for the linear predictor. The default is "link", unless the "predict.type" option is in force; see lsm.options.
df	If non-missing a constant number of degrees of freedom to use in constructing confidence intervals and P values (NA specifies asymptotic results).
null	Null hypothesis value(s) against which estimates are tested. May be a single value used for all, or a numeric vector of length equal to the number of tests in each family (i.e., by group in the displayed table).
delta	Numeric value. If zero, ordinary tests of significance are performed. If positive, this specifies a threshold for testing equivalence (using the TOST or two-one-sided-test method), non-inferiority, or non-superiority, depending on side. See Details for how the test statistics are defined.
side	Numeric or character value specifying whether the test is left-tailed (-1, "-", code"<", "left", or "nonsuperiority"); right-tailed (1, "+", ">", "right", or

	"noninferiority"); or two-sided (0, 2, "!=", "two-sided", "both", "equivalence", or "=").
x	The object to be printed, plotted, or converted.
y	This argument is ignored.
horizontal	Determines orientation of plotted confidence intervals.
intervals	If TRUE, confidence intervals are plotted for each estimate
comparisons	If TRUE, "comparison arrows" are added to the plot, in such a way that the degree to which arrows overlap reflects as much as possible the significance of the comparison of the two estimates.
alpha, int.adjust	
	The alpha argument to use in constructing comparison arrows. int.adjust may be used to set the adjust argument for the confidence intervals (use adjust to set the adjust method for the comparison arrows).
transform	Logical value; if true, the inverse transformation is applied to the estimates in the grid
names	Logical scalar or vector specifying whether variable names are appended to levels in the column labels for the as.mcmc result – e.g., column names of treat A and treat B versus just A and B. When there is more than one variable involved, the elements of names are used cyclically.
..., digits, quote, right, xlab, ylab, layout	
	For summaries, these are additional arguments passed to other methods including [print.data.frame], [update], or [dotplot] as appropriate. If not specified, appropriate defaults are used. For example, the default layout is one column of horizontal panels or one row of vertical panels.

Details

Defaults: The misc slot in object contains default values for by, infer, level, adjust, and type. These defaults vary depending on the code that created the object. The [update] method may be used to change these defaults. In addition, any options set using 'lsm.options(summary=...)' will trump those stored in the object's misc slot.

Transformations and links: With type="response", the transformation assumed can be found in 'object@misc$tran', and its label, for the summary is in 'object@misc$inv.lbl'. At this time, tran must be one of the named transformations valid for [make.link]. Any t or z tests are still performed on the scale of the linear predictor, not the inverse-transformed one. Similarly, confidence intervals are computed on the linear-predictor scale, then inverse-transformed.

Confidence-limit and P-value adjustments: The adjust argument has the following effects:

"tukey" Uses the Studentized range distribution with the number of means in the family. (Available for two-sided cases only.)

"scheffe" Computes p values from the F distribution, according to the Scheffe critical value of $\sqrt{kF(k, d)}$, where d is the error degrees of freedom and k is (family size minus 1) for contrasts, and (number of estimates) otherwise. (Available for two-sided cases only.)

"sidak" Makes adjustments as if the estimates were independent (a conservative adjustment in many cases).

"bonferroni" Multiplies p values, or divides significance levels by the number of estimates. This is a conservative adjustment.

"dunnettx" Uses an approximation to the Dunnett distribution for a family of estimates having pairwise correlations of 0.5 (as is true when comparing treatments with a control with equal sample sizes). The accuracy of the approximation improves with the number of simultaneous estimates, and is much faster than "mvt". (Available for two-sided cases only.)

"mvt" Uses the multivariate t distribution to assess the probability or critical value for the maximum of k estimates. This method produces the same p values and intervals as the default summary or confint methods to the results of as.glht. In the context of pairwise comparisons or comparisons with a control, this produces "exact" Tukey or Dunnett adjustments, respectively. However, the algorithm (from the **mvtnorm** package) uses a Monte Carlo method, so results are not exactly repeatable unless the random-number seed is used (see set.seed). As the family size increases, the required computation time will become noticeable or even intolerable, making the "tukey", "dunnettx", or others more attractive.

"none" Makes no adjustments to the p values.

For P-value adjustments only, the other Bonferroni-inequality-based adjustment methods in p.adjust are also available. If a p.adjust.methods method is specified for confidence limits, they are left unadjusted and an annotation is added that says so. Also, if an adjustment method is not appropriate (e.g., using "tukey" with one-sided tests, or with results that are not pairwise comparisons), a more appropriate method (usually "sidak") is substituted.

In some cases, confidence and p-value adjustments are only approximate – especially when the degrees of freedom or standard errors vary greatly within the family of tests. The "mvt" method is always the correct one-step adjustment, but it can be very slow. One may use as.glht with methods in the **multcomp** package to obtain non-conservative multi-step adjustments to tests.

Non-estimable cases: When the model is rank-deficient, each row x of object's linfct slot is each checked for estimability. If sum(x*bhat) is found to be non-estimable, then an NA is displayed for the estimate (as well as any associated statistics). This check is performed using the orthonormal basis N in the nbasis slot for the null space of the rows of the model matrix. Estimability fails when $||Nx||^2/||x||^2$ exceeds tol, which by default is 1e-8. You may change it via lsm.options by setting estble.tol to the desired value.

More on tests: When delta = 0, test statistics are of the usual form '(estimate - null)/SE', or notationally, $t = (Q - \theta_0)/SE$ where Q is our estimate of θ; then left, right, or two-sided p values are produced.

When delta is positive, the test statistic depends on side as follows.

Left-sided (nonsuperiority, $H_0 : \theta \geq \theta_0 + \delta$ versus $H_1 : \theta < \theta_0 + \delta$): $t = (Q - \theta_0 - \delta)/SE$. The p value is the lower-tail probability.

Right-sided (noninferiority): $H_0 : \theta \leq \theta_0 - \delta$ versus $H_1 : \theta > \theta_0 - \delta$): $t = (Q - \theta_0 + \delta)/SE$. The p value is the upper-tail probability.

Two-sided (equivalence): $H_0 : |\theta - \theta_0| \geq \delta$ versus $H_1 : |\theta - \theta_0| < \delta$): $t = (|Q - \theta_0| - \delta)/SE$. The p value is the *lower*-tail probability.

Plots: The plot method for "lsmobj" or "summary.ref.grid" objects (but not "ref.grid" objects themselves) produces a plot displaying confidence intervals for the estimates. If any by variables are in force, the plot is divided into separate panels. These functions use the dotplot function, and thus require that the **lattice** package be installed. For "summary.ref.grid" objects, the ...

arguments in `plot` are passed *only* to `dotplot`, whereas for `"lsmobj"` objects, the object is updated using `...` before summarizing and plotting.

In plots with `comparisons = TRUE`, the resulting arrows are only approximate, and in some cases may fail to accurately reflect the pairwise comparisons of the estimates – especially when estimates having large and small standard errors are intermingled in just the wrong way.

Re-gridding: The `regrid` function reparameterizes an existing `ref.grid` so that its `linfct` slot is the identity matrix and its `bhat` slot consists of the estimates at the grid points. If `transform` is `TRUE`, the inverse transform is applied to the estimates. Outwardly, the `summary` after applying `regrid` is identical to what it was before (using 'type="response"' if `transform` is TRUE). But subsequent contrasts will be conducted on the transformed scale – which is the reason this function exists. See the example below. **Warning:** in cases where degrees of freedom depends on the linear function being estimated, `regrid` will probably cause fatal errors when `summary` is called.

MCMC samplers: When the object's `post.beta` slot is non-trivial, `as.mcmc` will return an [mcmc](#) object that can be summarized or plotted using methods in the **coda** package. Specifically, `post.beta` is transformed by post-multiplying by `t(linfct)`, creating a sample from the posterior distribution of LS means.

Value

The `summary` method for `"ref.grid"` objects returns an object of class `"summary.ref.grid"`, which extends `"data.frame"`. `plot` returns an object of class `"trellis"`. `vcov` returns the covariance matrix of the product of the object's `linfct` and `bhat` slots. `as.mcmc` returns a **coda** mcmc object.

Author(s)

Russell V. Lenth

See Also

Methods for the closely related `"lsmobj"` class can be found in [contrast](#), [cld](#), and [glht](#). Also, [test](#) and [confint](#) are essentially front-ends for `summary`, so additional examples may be found there.

Examples

```
require(lsmeans)
warp.lm <- lm(breaks ~ wool * tension, data = warpbreaks)
warp.rg <- ref.grid(warp.lm)
str(warp.rg)

summary(warp.rg)

summary(warp.rg, by = "wool",
        infer = c(TRUE, FALSE), level = .90, adjust = "sidak")

# Transformed response
sqwarp.rg <- ref.grid(update(warp.lm, sqrt(breaks) ~ .))
summary(sqwarp.rg)
```

```
# Back-transformed results - compare with summary of 'warp.rg'
summary(sqwarp.rg, type = "response")

# But differences of sqrts can't be back-transformed
summary(pairs(sqwarp.rg, by = "wool"), type = "response")

# We can do it via regrid
sqwarp.rg2 <- regrid(sqwarp.rg)
summary(sqwarp.rg2)  # same as for sqwarp.rg with type = "response"
pairs(sqwarp.rg2, by = "wool")

# Logistic regression
# Reshape the Titanic data
Titan <- do.call("expand.grid", dimnames(Titanic)[-4])
Titan$Died <- matrix(Titanic, ncol=2)
Titan.glm <- glm(Died ~ (Class + Sex + Age)^2,
    family = binomial, data = Titan)
Titan.lsm <- lsmeans(Titan.glm, ~ Class|Sex, at = list(Age="Adult"))
summary(Titan.lsm, type="response")
summary(pairs(Titan.lsm), type="response")

# Nonsuperiority test: Is any class no more likely to die than
# the 1st class passengers?
summary(contrast(Titan.lsm, "trt.vs.ctrl1"), delta = 1,
    adjust = "none", side = "<")


# Plot 90% CIs on the response scale
plot(Titan.lsm, type = "response", level = .90,
     xlab = "Predicted probability of drowning")
```

---

update   *Set options for* ref.grid *or* lsmobj *objects*

Description

Objects of class ref.grid or lsmobj contain several settings in their "misc" slot that affect primarily the defaults used by summary. This update method allows them to be changed more safely than by modifying this slot directly.

In addition, the user may set defaults for all objects using 'options(lsmeans = ...)', or more conveniently using the lsm.options function documented here.

Usage

```
## S3 method for class 'ref.grid'
update(object, ..., silent = FALSE)

lsm.options(...)
```

Arguments

object	An object of class ref.grid (or its extension, lsmobj)
...	Arguments specifying elements' names and their new values.
silent	If FALSE, a message is displayed for any unmatched names.

Details

In update, the names in ... are partially matched against those that are valid, and if a match is found, it adds or replaces the current setting. The valid names are

tran (list or character) specifies the transformation which, when inverted, determines the results displayed by [summary](#), [predict](#), or [lsmip](#) when type="response". The value may be the name of a standard transformation from [make.link](#), or, for a custom transformation, a list containing at least the functions linkinv (the inverse of the transformation) and mu.eta (the derivative thereof). See the Examples.

estName (character) is the column label used for displaying predictions or LS means.

inv.lbl (character)) is the column label to use for predictions or LS means when type="response".

by.vars (character) vector or NULL) the variables used for grouping in the summary, and also for defining subfamilies in a call to [contrast](#).

pri.vars (character vector) are the names of the grid variables that are not in by.vars. Thus, the combinations of their levels are used as columns in each table produced by [summary](#).

alpha (numeric) is the default significance level for tests, in [summary](#) as well as [cld](#) and [plot](#) when 'intervals = TRUE'

adjust (character)) is the default for the adjust argument in [summary](#).

estType (character) is the type of the estimate. It should match one of 'c("prediction","contrast","pairs")'. This is used along with "adjust" to determine appropriate adjustments to P values and confidence intervals.

famSize (integer) is the nmeans parameter for [ptukey](#) when adjust="tukey".

infer (logical vector of length 2) is the default value of infer in [summary](#).

level (numeric) is the default confidence level, level, in [summary](#)

df (numeric) overrides the default degrees of freedom with a specified single value.

predict.type (character) sets the default method of displaying predictions in [summary](#), [predict](#), and [lsmip](#). Valid values are "link" (with synonyms "lp" and "linear"), or "response".

avgd.over (character) vector) are the names of the variables whose levels are averaged over in obtaining marginal averages of predictions, i.e., LS means. Changing this might produce a misleading printout, but setting it to character(0) will suppress the "averaged over" message in the summary.

initMesg (character) is a string that is added to the beginning of any annotations that appear below the [summary](#) display.

methDesc (character) is a string that may be used for creating names for a list of lsmobj objects.

In lsm.options, we may set or change the default values for the above attributes in the lsmeans option list(see [options](#)). Currently, the following elements of this list are used if specified:

ref.grid A named `list` of defaults for objects created by [ref.grid](#). This could affect other
    iobjects as well. For example, if lsmeans is called with a fitted model object, it calls `ref.grid`
    and this option will affect the resulting `lsmobj` object.

lsmeans A named `list` of defaults for objects created by [lsmeans](#) (or [lstrends](#)).

contrast A named `list` of defaults for objects created by [contrast](#) (or [pairs](#)).

summary A named `list` of defaults used by the methods [summary](#), [predict](#), and [lsmip](#). The only
    option that can affect the latter two is `"predict.method"`.

estble.tol Tolerance for determining estimability in rank-deficient cases. If absent, `1e-8` is used.

Value

update returns a copy of `object` with its `"misc"` slot modified. `lsm.options` returns the current
options (same as the result of 'getOption("lsmeans")').

Note

If a call to [lsmeans](#), [contrast](#), or [ref.grid](#) contains a non-NULL `options` list, those options are
passed in a call to update on the constructed object before it is returned. This allows you, for
example, to override the defaults used by [summary](#). In addition, user defaults may be set using
an `link{options}` setting for `"lsmeans"`. It should be a list with one or more named elements
lsmeans, contrast, or ref.grid, used for setting the defaults for objects constructed by functions
of these same names. Note that options can get "inherited". See the examples.

Unlike the update method for model classes (lm, glm, etc.), this does not re-fit or re-estimate
anything; but it does affect how `object` is treated by other methods for its class.

Author(s)

Russell V. Lenth

See Also

[summary](#)

Examples

```
# An altered log transformation

warp.lm1 <- lm(log(breaks + 1) ~ wool*tension, data = warpbreaks)
rg1 <- update(ref.grid(warp.lm1),
        tran = list(linkinv = function(eta) exp(eta) - 1,
                    mu.eta = function(eta) exp(eta)),
        inv.lbl = "pred.breaks")

summary(rg1, type = "response")

## Not run:
lsm.options(ref.grid = list(level = .90),
            contrast = list(infer = c(TRUE,FALSE)),
            estble.tol = 1e-6)
```

```
# Sets default confidence level to .90 for objects created by ref.grid
# AS WELL AS lsmeans called with a model object (since it creates a
# reference grid). In addition, when we call 'contrast', 'pairs', etc.,
# confidence intervals rather than tests are displayed by default.

lsm.options(disable.pbkrtest = TRUE)
# This forces use of asymptotic methods for lmerMod objects.
# Set to FALSE or NULL to re-enable using pbkrtest.

print(lsm.options())    # see the current settings

## End(Not run)
```

Index