Package ‘lmerTest’

March 5, 2026

Type Package

Title Tests in Linear Mixed Effects Models

Version 3.2-1

Depends R (>= 3.2.5), lme4 (>= 1.1-10), stats, methods

Imports numDeriv, MASS, ggplot2, reformulas

Suggests pbkrtest (>= 0.4-3), tools

Description Provides p-values in type I, II or III anova and summary tables for lmer model fits (cf. lme4) via Satterthwaite's degrees of freedom method. A Kenward-Roger method is also available via the pbkrtest package. Model selection methods include step, drop1 and anova-like tables for random effects (ranova). Methods for Least-Square means (LS-means) and tests of linear contrasts of fixed effects are also available.

License GPL (>= 2)

Encoding UTF-8

LazyData true

URL https://github.com/runehaubo/lmerTestR

BugReports https://github.com/runehaubo/lmerTestR/issues

RoxygenNote 7.3.3

Collate 'anova_contrasts.R' 'contest.R' 'contrast_utils.R' 'data_documentation.R' 'drop1.R' 'estimability.R' 'legacy.R' 'lmer.R' 'lmerTest.R' 'lmer_anova.R' 'lmer_summary.R' 'ls_means.R' 'ranova.R' 'step.R' 'terms_utils.R' 'utils.R'

NeedsCompilation no

Author Alexandra Kuznetsova [aut],
Per Bruun Brockhoff [aut, ths],
Rune Haubo Bojesen Christensen [aut, cre] (ORCID:
<https://orcid.org/0000-0002-4494-3399>),
Sofie Pødenphant Jensen [ctb]

Maintainer Rune Haubo Bojesen Christensen <Rune.Haubo@gmail.com>

Repository CRAN

Date/Publication 2026-03-05 12:20:02 UTC

Contents

`anova.lmerModLmerTest` *ANOVA Tables for Linear Mixed Models*

Description

ANOVA table with F-tests and p-values using Satterthwaite's or Kenward-Roger's method for denominator degrees-of-freedom and F-statistic. Models should be fitted with `lmer` from the **lmerTest**-package.

Usage

```
## S3 method for class 'lmerModLmerTest'
anova(
  object,
  ...,
  type = c("III", "II", "I", "3", "2", "1"),
  ddf = c("Satterthwaite", "Kenward-Roger", "lme4")
)
```

Arguments

`object`	an `lmerModLmerTest` object; the result of `lmer()` after loading the **lmerTest**-package.
`...`	potentially additional `lmer` or `lm` model objects for comparison of models in which case `type` and `ddf` arguments are ignored.

type — the type of ANOVA table requested (using SAS terminology) with Type I being the familiar sequential ANOVA table.

ddf — the method for computing the denominator degrees of freedom and F-statistics. `ddf="Satterthwaite"` (default) uses Satterthwaite's method; `ddf="Kenward-Roger"` uses Kenward-Roger's method, `ddf = "lme4"` returns the lme4-anova table, i.e., using the anova method for `lmerMod` objects as defined in the **lme4**-package and ignores the `type` argument. Partial matching is allowed.

Details

The `"Kenward-Roger"` method calls `pbkrtest::KRmodcomp` internally and reports scaled F-statistics and associated denominator degrees-of-freedom.

Value

an ANOVA table

Author(s)

Rune Haubo B. Christensen and Alexandra Kuznetsova

See Also

`contestMD` for multi degree-of-freedom contrast tests and `KRmodcomp` for the `"Kenward-Roger"` method.

Examples

```
data("sleepstudy", package="lme4")
m <- lmer(Reaction ~ Days + (Days | Subject), sleepstudy)
anova(m) # with p-values from F-tests using Satterthwaite's denominator df
anova(m, ddf="lme4") # no p-values

# Use the Kenward-Roger method
if(requireNamespace("pbkrtest", quietly = TRUE))
  anova(m, ddf="Kenward-Roger")
```

`as_lmerModLmerTest` *Coerce lmerMod Objects to lmerModLmerTest*

Description

Coercing an lme4::lmer model-object (of class 'lmerMod') to a model-object of class 'lmerModLmerTest' involves computing the covariance matrix of the variance parameters and the gradient (Jacobian) of cov(beta) with respect to the variance parameters.

Usage

```
as_lmerModLmerTest(model, tol = 1e-08)
```

Arguments

`model`	and lmer model-object (of class 'lmerMod') – the result of a call to `lme4::lmer()`
`tol`	tolerance for determining of eigenvalues are negative, zero or positive

Value

an object of class `'lmerModLmerTest'` which sets the following slots:

`vcov_varpar`	the asymptotic covariance matrix of the variance parameters (theta, sigma).
`Jac_list`	list of Jacobian matrices; gradients of vcov(beta) with respect to the variance parameters.
`vcov_beta`	the asymptotic covariance matrix of the fixed-effect regression parameters (beta; vcov(beta)).
`sigma`	the residual standard deviation.

Author(s)

Rune Haubo B. Christensen

See Also

the class definition in `lmerModLmerTest`) and `lmer`

Examples

```
m <- lme4::lmer(Reaction ~ Days + (Days | Subject), sleepstudy)
bm <- as_lmerModLmerTest(m)
slotNames(bm)
```

`carrots` *Consumer Preference Mapping of Carrots*

Description

In a consumer study 103 consumers scored their preference of 12 danish carrot types on a scale from 1 to 7. Moreover the consumers scored the degree of sweetness, bitterness and crispiness in the products.

Usage

```
data(carrots)
```

Format

Consumer factor with 103 levels: numbering identifying consumers.

Frequency factor with 5 levels; "How often do you eat carrots?" 1: once a week or more, 2: once every two weeks, 3: once every three weeks, 4: at least once month, 5: less than once a month.

Gender factor with 2 levels. 1: male, 2:female.

Age factor with 4 levels. 1: less than 25 years, 2: 26-40 years, 3: 41-60 years, 4 more than 61 years.

Homesize factor with two levels. Number of persons in the household. 1: 1 or 2 persons, 2: 3 or more persons.

Work factor with 7 levels. different types of employment. 1: unskilled worker(no education), 2: skilled worker(with education), 3: office worker, 4: housewife (or man), 5: independent businessman/ self-employment, 6: student, 7: retired

Income factor with 4 levels. 1: <150000, 2: 150000-300000, 3: 300000-500000, 4: >500000

Preference consumer score on a seven-point scale.

Sweetness consumer score on a seven-point scale.

Bitterness consumer score on a seven-point scale.

Crispness consumer score on a seven-point scale.

sens1 first sensory variable derived from a PCA.

sens2 second sensory variable derived from a PCA.

Product factor on 12 levels.

Details

The carrots were harvested in autumn 1996 and tested in march 1997. In addition to the consumer survey, the carrot products were evaluated by a trained panel of tasters, the sensory panel, with respect to a number of sensory (taste, odour and texture) properties. Since usually a high number of (correlated) properties (variables) are used, in this case 14, it is a common procedure to use a few, often 2, combined variables that contain as much of the information in the sensory variables as possible. This is achieved by extracting the first two principal components in a principal components analysis (PCA) on the product-by-property panel average data matrix. In this data set the variables for the first two principal components are named (`sens1` and `sens2`).

Source

Per Bruun Brockhoff, The Royal Veterinary and Agricultural University, Denmark.

Examples

```
fm <- lmer(Preference ~ sens2 + Homesize + (1 + sens2 | Consumer), data=carrots)
anova(fm)
```

`contest.lmerModLmerTest` *Test of Contrasts*

Description

Tests of vector or matrix contrasts for `lmer` model fits.

Usage

```
## S3 method for class 'lmerModLmerTest'
contest(
  model,
  L,
  rhs = 0,
  joint = TRUE,
  collect = TRUE,
  confint = TRUE,
  level = 0.95,
  check_estimability = FALSE,
  ddf = c("Satterthwaite", "Kenward-Roger", "lme4"),
  ...
)

## S3 method for class 'lmerMod'
contest(
  model,
  L,
  rhs = 0,
  joint = TRUE,
  collect = TRUE,
  confint = TRUE,
  level = 0.95,
  check_estimability = FALSE,
  ddf = c("Satterthwaite", "Kenward-Roger", "lme4"),
  ...
)
```

Arguments

`model`	a model object fitted with `lmer` from package **lmerTest**, i.e., an object of class `lmerModLmerTest`.
`L`	a contrast vector or matrix or a list of these. The `length`/`ncol` of each contrasts should equal `length(fixef(model))`.
`rhs`	right-hand-side of the statistical test, i.e. the hypothesized value (a numeric scalar).

joint	make an F-test of potentially several contrast vectors? If FALSE single DF t-tests are applied to each vector or each row of contrasts matrices.
collect	collect list of tests in a matrix?
confint	include columns for lower and upper confidence limits? Applies when joint is FALSE.
level	confidence level.
check_estimability	check estimability of contrasts? Only single DF contrasts are checked for estimability thus requiring joint = FALSE to take effect. See details section for necessary adjustments to L when estimability is checked with rank deficient design matrices.
ddf	the method for computing the denominator degrees of freedom. ddf="Kenward-Roger" uses Kenward-Roger's method.
...	passed to contestMD.

Details

If the design matrix is rank deficient, lmer drops columns for the aliased coefficients from the design matrix and excludes the corresponding aliased coefficients from fixef(model). When estimability is checked the original rank-deficient design matrix is recontructed and therefore L contrast vectors need to include elements for the aliased coefficients. Similarly when L is a matrix, its number of columns needs to match that of the reconstructed rank-deficient design matrix.

Value

a data.frame or a list of data.frames.

Author(s)

Rune Haubo B. Christensen

See Also

contestMD for multi degree-of-freedom contrast tests, and contest1D for tests of 1-dimensional contrasts.

Examples

```
data("sleepstudy", package="lme4")
fm <- lmer(Reaction ~ Days + I(Days^2) + (1|Subject) + (0+Days|Subject),
           sleepstudy)
# F-test of third coeffcients - I(Days^2):
contest(fm, c(0, 0, 1))
# Equivalent t-test:
contest(fm, L=c(0, 0, 1), joint=FALSE)
# Test of 'Days + I(Days^2)':
contest(fm, L=diag(3)[2:3, ])
# Other options:
contest(fm, L=diag(3)[2:3, ], joint=FALSE)
```

```
contest(fm, L=diag(3)[2:3, ], joint=FALSE, collect=FALSE)

# Illustrate a list argument:
L <- list("First"=diag(3)[3, ], "Second"=diag(3)[-1, ])
contest(fm, L)
contest(fm, L, collect = FALSE)
contest(fm, L, joint=FALSE, confint = FALSE)
contest(fm, L, joint=FALSE, collect = FALSE, level=0.99)

# Illustrate testing of estimability:
# Consider the 'cake' dataset with a missing cell:
data("cake", package="lme4")
cake$temperature <- factor(cake$temperature, ordered=FALSE)
cake <- droplevels(subset(cake, temperature %in% levels(cake$temperature)[1:2] &
                            !(recipe == "C" & temperature == "185")))
with(cake, table(recipe, temperature))
fm <- lmer(angle ~ recipe * temperature + (1|recipe:replicate), cake)
fixef(fm)
# The coefficient for recipeC:temperature185 is dropped:
attr(model.matrix(fm), "col.dropped")
# so any contrast involving this coefficient is not estimable:
Lmat <- diag(6)
contest(fm, Lmat, joint=FALSE, check_estimability = TRUE)
```

---

contest1D.lmerModLmerTest

Contrast Tests in 1D

Description

Compute the test of a one-dimensional (vector) contrast in a linear mixed model fitted with lmer from package **lmerTest**. The contrast should specify a linear function of the mean-value parameters, beta. The Satterthwaite or Kenward-Roger method is used to compute the (denominator) df for the t-test.

Usage

```
## S3 method for class 'lmerModLmerTest'
contest1D(
  model,
  L,
  rhs = 0,
  ddf = c("Satterthwaite", "Kenward-Roger"),
  confint = FALSE,
  level = 0.95,
  ...
)
```

```
## S3 method for class 'lmerMod'
contest1D(
  model,
  L,
  rhs = 0,
  ddf = c("Satterthwaite", "Kenward-Roger"),
  confint = FALSE,
  level = 0.95,
  ...
)
```

Arguments

model	a model object fitted with lmer from package **lmerTest**, i.e., an object of class lmerModLmerTest.
L	a numeric (contrast) vector of the same length as fixef(model).
rhs	right-hand-side of the statistical test, i.e. the hypothesized value (a numeric scalar).
ddf	the method for computing the denominator degrees of freedom. ddf="Kenward-Roger" uses Kenward-Roger's method.
confint	include columns for lower and upper confidence limits?
level	confidence level.
...	currently not used.

Details

The t-value and associated p-value is for the hypothesis $L'\beta = \mathrm{rhs}$ in which rhs may be non-zero and β is fixef(model). The estimated value ("Estimate") is $L'\beta$ with associated standard error and (optionally) confidence interval.

Value

A data.frame with one row and columns with "Estimate", "Std. Error", "t value", "df", and "Pr(>|t|)" (p-value). If confint = TRUE "lower" and "upper" columns are included before the p-value column.

Author(s)

Rune Haubo B. Christensen

See Also

contest for a flexible and general interface to tests of contrasts among fixed-effect parameters. contestMD is also available as a direct interface for tests of multi degree-of-freedom contrast.

Examples

```
# Fit model using lmer with data from the lme4-package:
data("sleepstudy", package="lme4")
fm <- lmer(Reaction ~ Days + (1 + Days|Subject), sleepstudy)

# Tests and CI of model coefficients are obtained with:
contest1D(fm, c(1, 0), confint=TRUE) # Test for Intercept
contest1D(fm, c(0, 1), confint=TRUE) # Test for Days

# Tests of coefficients are also part of:
summary(fm)

# Illustrate use of rhs argument:
contest1D(fm, c(0, 1), confint=TRUE, rhs=10) # Test for Days-coef == 10
```

---

contestMD.lmerModLmerTest

Multiple Degrees-of-Freedom Contrast Tests

Description

Compute the multi degrees-of-freedom test in a linear mixed model fitted by `lmer`. The contrast (L) specifies a linear function of the mean-value parameters, beta. Satterthwaite's method is used to compute the denominator df for the F-test.

Usage

```
## S3 method for class 'lmerModLmerTest'
contestMD(
  model,
  L,
  rhs = 0,
  ddf = c("Satterthwaite", "Kenward-Roger"),
  eps = sqrt(.Machine$double.eps),
  ...
)

calcSatterth(model, L)

## S3 method for class 'lmerMod'
contestMD(
  model,
  L,
  rhs = 0,
  ddf = c("Satterthwaite", "Kenward-Roger"),
  eps = sqrt(.Machine$double.eps),
```

```
  ...
)
```

Arguments

`model`	a model object fitted with `lmer` from package **lmerTest**, i.e., an object of class `lmerModLmerTest`.
`L`	a contrast matrix with nrow >= 1 and ncol == `length(fixef(model))`.
`rhs`	right-hand-side of the statistical test, i.e. the hypothesized value. A numeric vector of length `nrow(L)` or a numeric scalar.
`ddf`	the method for computing the denominator degrees of freedom and F-statistics. `ddf="Kenward-Roger"` uses Kenward-Roger's method.
`eps`	tolerance on eigenvalues to determine if an eigenvalue is positive. The number of positive eigenvalues determine the rank of L and the numerator df of the F-test.
`...`	currently not used.

Details

The F-value and associated p-value is for the hypothesis $L\beta = \mathrm{rhs}$ in which rhs may be non-zero and β is `fixef(model)`.

Note: NumDF = row-rank(L) is determined automatically so row rank-deficient L are allowed. One-dimensional contrasts are also allowed (L has 1 row).

Value

a `data.frame` with one row and columns with `"Sum Sq"`, `"Mean Sq"`, `"F value"`, `"NumDF"` (numerator df), `"DenDF"` (denominator df) and `"Pr(>F)"` (p-value).

Author(s)

Rune Haubo B. Christensen

See Also

`contest` for a flexible and general interface to tests of contrasts among fixed-effect parameters. `contest1D` is a direct interface for tests of 1-dimensional contrasts.

Examples

```
data("sleepstudy", package="lme4")
fm <- lmer(Reaction ~ Days + I(Days^2) + (1|Subject) + (0+Days|Subject),
           sleepstudy)

# Define 2-df contrast - since L has 2 (linearly independent) rows
# the F-test is on 2 (numerator) df:
L <- rbind(c(0, 1, 0), # Note: ncol(L) == length(fixef(fm))
           c(0, 0, 1))
```

```
# Make the 2-df F-test of any effect of Days:
contestMD(fm, L)

# Illustrate rhs argument:
contestMD(fm, L, rhs=c(5, .1))

# Make the 1-df F-test of the effect of Days^2:
contestMD(fm, L[2, , drop=FALSE])
# Same test, but now as a t-test instead:
contest1D(fm, L[2, , drop=TRUE])
```

---

drop1.lmerModLmerTest *Drop Marginal Terms from Model*

Description

Computes the F-test for all marginal terms, i.e. terms that can be dropped from the model while respecting the hierarchy of terms in the model.

Usage

```
## S3 method for class 'lmerModLmerTest'
drop1(
  object,
  scope,
  ddf = c("Satterthwaite", "Kenward-Roger", "lme4"),
  force_get_contrasts = FALSE,
  ...
)
```

Arguments

object	an `lmer` model fit (of class "lmerModLmerTest".)
scope	optional character vector naming terms to be dropped from the model. Note that only marginal terms can be dropped. To see which terms are marginal, use `drop.scope(terms(object))`.
ddf	the method for computing the denominator degrees of freedom and F-statistics. `ddf="Satterthwaite"` (default) uses Satterthwaite's method; `ddf="Kenward-Roger"` uses Kenward-Roger's method. `ddf = "lme4"` returns the `drop1` table for `merMod` objects as defined in package **lme4**.
force_get_contrasts	enforce computation of contrast matrices by a method in which the design matrices for full and restricted models are compared.
...	currently not used.

Details

Simple marginal contrasts are used for all marginal terms unless the design matrix is rank deficient. In that case (and if `force_get_contrasts` is `TRUE`) the contrasts (i.e. restriction matrices on the design matrix of the full model) are computed by comparison of the design matrices for full and restricted models. The set of marginal terms considered for dropping are computed using `drop.scope(terms(object))`.

Since all tests are based on tests of contrasts in the full model, no models are being (re)fitted.

Value

An anova-like table with F-tests of marginal terms.

Author(s)

Rune Haubo B. Christensen

See Also

`ranova` for tests of marginal random terms.

Examples

```
# Basic usage:
fm <- lmer(angle ~ recipe + temp + (1|recipe:replicate), cake)
drop1(fm) # Using Satterthwaite degrees of freedom
if(requireNamespace("pbkrtest", quietly = TRUE))
  drop1(fm, ddf="Kenward-Roger") # Alternative DenDF and F-test method
drop1(fm, ddf="lme4", test="Chi") # Asymptotic Likelihood ratio tests

# Consider a rank-deficient design matrix:
fm <- lmer(angle ~ recipe + temp + temperature + (1|recipe:replicate), cake)
# Here temp accounts for the linear effect of temperature, and
# temperature is an (ordered) factor that accounts for the remaining
# variation between temperatures (4 df).
drop1(fm)
# While temperature is in the model, we cannot test the effect of dropping
# temp. After removing temperature we can test the effect of dropping temp:
drop1(lmer(angle ~ recipe + temp + (1|recipe:replicate), cake))

# Polynomials:
# Note that linear terms should usually not be dropped before squared terms.
# Therefore 'Days' should not be dropped before 'I(Days^2)' despite it being
# tested here:
fm <- lmer(Reaction ~ Days + I(Days^2) + (Days|Subject), sleepstudy)
drop1(fm)
# Using poly() provides a test of the whole polynomial structure - not a
# separate test for the highest order (squared) term:
fm <- lmer(Reaction ~ poly(Days, 2) + (Days|Subject), sleepstudy)
drop1(fm)
```

ham *Conjoint Study of Dry Cured Ham*

Description

One of the purposes of the study was to investigate the effect of information given to the consumers measured in hedonic liking for the hams. Two of the hams were Spanish and two were Norwegian, each origin representing different salt levels and different aging time. The information about origin was given in such way that both true and false information was given. Essentially a 4x2 design with 4 samples and 2 information levels. A total of 81 Consumers participated in the study.

Usage

```
data(ham)
```

Format

Consumer factor with 81 levels: numbering identifying consumers.

Product factor with four levels.

Informed.liking numeric: hedonic liking for the products.

Information factor with two levels.

Gender factor with two levels.

Age numeric: age of Consumer.

References

T. Næs, V. Lengard, S. Bølling Johansen, M. Hersleth (2010) Alternative methods for combining design variables and consumer preference with information about attitudes and demographics in conjoint analysis, *Food Quality and Preference*, 10-4, 368-378, ISSN 0950-3293, doi:10.1016/j.foodqual.2009.09.004.

Examples

```
# Simple model for the ham data:
fm <- lmer(Informed.liking ~ Product*Information + (1|Consumer) , data=ham)

# Anova table for the fixed effects:
anova(fm)

## Not run:
# Fit 'big' model:
fm <- lmer(Informed.liking ~ Product*Information*Gender*Age +
             + (1|Consumer) + (1|Consumer:Product) +
             (1|Consumer:Information),
           data=ham)
step_fm <- step(fm)
step_fm # Display elimination results
```

```
final_fm <- get_model(step_fm)

## End(Not run)
```

---

lmer *Fit Linear Mixed-Effects Models*

Description

This function overloads `lmer` from the **lme4**-package (`lme4::lmer`) and adds a couple of slots needed for the computation of Satterthwaite denominator degrees of freedom. All arguments are the same as for `lme4::lmer` and all the usual `lmer`-methods work.

Usage

```
lmer(
  formula,
  data = NULL,
  REML = TRUE,
  control = lmerControl(),
  start = NULL,
  verbose = 0L,
  subset,
  weights,
  na.action,
  offset,
  contrasts = NULL,
  devFunOnly = FALSE
)
```

Arguments

`formula` a two-sided linear formula object describing both the fixed-effects and random-effects part of the model, with the response on the left of a ~ operator and the terms, separated by + operators, on the right. Random-effects terms are distinguished by vertical bars (`|`) separating expressions for design matrices from grouping factors. By default, non-scalar random effects (where the design matrix has more than one column, e.g. `(1+x|f)`) are fitted with *unstructured* (general positive semidefinite) covariance matrices.

- Two vertical bars (`||`) can be used to specify multiple uncorrelated random effects for the same grouping variable. With default settings, the `||`-syntax *works only for design matrices containing numeric (continuous) predictors*; to fit models with independent categorical effects, use `diag(f|g)` or set `options(lme4.doublevert.default = "diag_special")` (see `help(getDoublevertDefault)` in **lme4** version `2.0-0` or higher).
- Tags preceding a random effect term specify covariance structure:

- us (default: `us(f|g)` is equivalent to `(f|g)`): unstructured, positive semi-definite
- `diag`: diagonal (all correlations set to zero). Specify `diag(f|g, hom = TRUE)` to fit a homogeneous diagonal covariance matrix
- `cs`: compound symmetric (all pairwise correlations set identical). Specify `cs(f|g, hom = TRUE)` for homogeneous variances.
- `ar1`: autoregressive order 1. Note that AR1 models are homogeneous by default; specify `ar1(f|g, hom = FALSE)` for heterogeneous variances.

`data` an optional data frame containing the variables named in `formula`. By default the variables are taken from the environment from which `lmer` is called. While `data` is optional, the package authors *strongly* recommend its use, especially when later applying methods such as `update` and `drop1` to the fitted model (*such methods are not guaranteed to work properly if* `data` *is omitted*). If `data` is omitted, variables will be taken from the environment of `formula` (if specified as a formula) or from the parent frame (if specified as a character vector).

`REML` logical scalar - Should the estimates be chosen to optimize the REML criterion (as opposed to the log-likelihood)?

`control` a list (of correct class, resulting from `lmerControl()` or `glmerControl()` respectively) containing control parameters, including the nonlinear optimizer to be used and parameters to be passed through to the nonlinear optimizer, see the `*lmerControl` documentation for details.

`start` a numeric vector or a named list with one optional component named `par` or `theta`, giving starting values for covariance parameters. Numeric `start` is equivalent to `list(par = start)`. Parameters corresponding to unstructured covariance matrices are on the scale of the Cholesky factor of the relative covariance matrix. By default, all relative covariance matrices are identity matrices.

`verbose` integer scalar. If `> 0` verbose output is generated during the optimization of the parameter estimates. If `> 1` verbose output is generated during the individual penalized iteratively reweighted least squares (PIRLS) steps.

`subset` an optional expression indicating the subset of the rows of `data` that should be used in the fit. This can be a logical vector, or a numeric vector indicating which observation numbers are to be included, or a character vector of the row names to be included. All observations are included by default.

`weights` an optional vector of 'prior weights' to be used in the fitting process. Should be `NULL` or a numeric vector. Prior `weights` are *not* normalized or standardized in any way. In particular, the diagonal of the residual covariance matrix is the squared residual standard deviation parameter `sigma` times the vector of inverse `weights`. Therefore, if the `weights` have relatively large magnitudes, then in order to compensate, the `sigma` parameter will also need to have a relatively large magnitude.

`na.action` a function that indicates what should happen when the data contain `NA`s. The default action (`na.omit`, inherited from the 'factory fresh' value of `getOption("na.action")`) strips any observations with any missing values in any variables.

`offset` this can be used to specify an *a priori* known component to be included in the linear predictor during fitting. This should be `NULL` or a numeric vector of length equal to the number of cases. One or more `offset` terms can be included in the formula instead or as well, and if more than one is specified their sum is used. See `model.offset`.

`contrasts` an optional list. See the `contrasts.arg` of `model.matrix.default`.

`devFunOnly` logical - return only the deviance evaluation function. Note that because the deviance function operates on variables stored in its environment, it may not return *exactly* the same values on subsequent calls (but the results should always be within machine tolerance).

Details

For details about `lmer` see `lmer` (`help(lme4::lmer)`). The description of all arguments below is taken verbatim and unedited from the **lme4**-package.

In cases when a valid `lmer`-object (`lmerMod`) is produced, but when the computations needed for Satterthwaite df fails, the `lmerMod` object is returned - not an `lmerModLmerTest` object.

Value

an S4 object of class `"lmerModLmerTest"`

Author(s)

Rune Haubo B. Christensen and Alexandra Kuznetsova for the overload in **lmerTest** – **lme4**-authors for the underlying implementation in **lme4**.

See Also

`lmer` and `lmerModLmerTest`

Examples

```
data("sleepstudy", package="lme4")
m <- lmer(Reaction ~ Days + (Days | Subject), sleepstudy)
class(m) # lmerModLmerTest
```

`lmerModLmerTest-class` *Represent Linear Mixed-Effects Models*

Description

The `lmerModLmerTest` class extends `lmerMod` (which extends `merMod`) from the **lme4**-package.

Value

An object of class `lmerModLmerTest` with slots as in `lmerMod` objects (see `merMod`) and a few additional slots as described in the slots section.

Slots

vcov_varpar a numeric matrix holding the asymptotic variance-covariance matrix of the variance parameters (including sigma).

Jac_list a list of gradient matrices (Jacobians) for the gradient of the variance-covariance of beta with respect to the variance parameters, where beta are the mean-value parameters available in fixef(object).

vcov_beta a numeric matrix holding the asymptotic variance-covariance matrix of the fixed-effect regression parameters (beta).

sigma the residual standard deviation.

Author(s)

Rune Haubo B. Christensen

See Also

lmer and merMod

---

ls_means.lmerModLmerTest

LS-means for lmerTest Model Fits

Description

Computes LS-means or pairwise differences of LS-mean for all factors in a linear mixed model. lsmeansLT is provided as an alias for ls_means for backward compatibility.

Usage

```
## S3 method for class 'lmerModLmerTest'
ls_means(
  model,
  which = NULL,
  level = 0.95,
  ddf = c("Satterthwaite", "Kenward-Roger"),
  pairwise = FALSE,
  ...
)

## S3 method for class 'lmerModLmerTest'
lsmeansLT(
  model,
  which = NULL,
  level = 0.95,
  ddf = c("Satterthwaite", "Kenward-Roger"),
  pairwise = FALSE,
```

```
  ...
)

## S3 method for class 'lmerModLmerTest'
difflsmeans(
  model,
  which = NULL,
  level = 0.95,
  ddf = c("Satterthwaite", "Kenward-Roger"),
  ...
)
```

Arguments

`model`	a model object fitted with `lmer` (of class `"lmerModLmerTest"`).
`which`	optional character vector naming factors for which LS-means should be computed. If `NULL` (default) LS-means for all factors are computed.
`level`	confidence level.
`ddf`	method for computation of denominator degrees of freedom.
`pairwise`	compute pairwise differences of LS-means instead?
`...`	currently not used.

Details

Confidence intervals and p-values are based on the t-distribution using degrees of freedom based on Satterthwaites or Kenward-Roger methods.

LS-means is SAS terminology for predicted/estimated marginal means, i.e. means for levels of factors which are averaged over the levels of other factors in the model. A flat (i.e. unweighted) average is taken which gives equal weight to all levels of each of the other factors. Numeric/continuous variables are set at their mean values. See **emmeans** package for more options and greater flexibility.

LS-means contrasts are checked for estimability and unestimable contrasts appear as `NA`s in the resulting table.

LS-means objects (of class `"ls_means"` have a print method).

Value

An LS-means table in the form of a `data.frame`. Formally an object of class `c("ls_means", "data.frame")` with a number of attributes set.

Author(s)

Rune Haubo B. Christensen and Alexandra Kuznetsova

See Also

`show_tests` for display of the underlying LS-means contrasts.

Examples

```
# Get data and fit model:
data("cake", package="lme4")
model <- lmer(angle ~ recipe * temp + (1|recipe:replicate), cake)

# Compute LS-means:
ls_means(model)

# Get LS-means contrasts:
show_tests(ls_means(model))

# Compute pairwise differences of LS-means for each factor:
ls_means(model, pairwise=TRUE)
difflsmeans(model) # Equivalent.
```

`ranova` *ANOVA-Like Table for Random-Effects*

Description

Compute an ANOVA-like table with tests of random-effect terms in the model. Each random-effect term is reduced or removed and likelihood ratio tests of model reductions are presented in a form similar to that of `drop1`. `rand` is an alias for `ranova`.

Usage

```
ranova(model, reduce.terms = TRUE, ...)

rand(model, reduce.terms = TRUE, ...)
```

Arguments

`model`	a linear mixed effect model fitted with `lmer()` (inheriting from class `lmerMod`).
`reduce.terms`	if `TRUE` (default) random-effect terms are reduced (if possible). If `FALSE` random-effect terms are simply removed.
`...`	currently ignored

Details

If the model is fitted with REML the tests are REML-likelihood ratio tests.

A random-effect term of the form `(f1 + f2 | gr)` is reduced to terms of the form `(f2 | gr)` and `(f1 | gr)` and these reduced models are compared to the original model. If `reduce.terms` is `FALSE` `(f1 + f2 | gr)` is removed instead.

A random-effect term of the form `(f1 | gr)` is reduced to `(1 | gr)` (unless `reduce.terms` is `FALSE`).

A random-effect term of the form `(1 | gr)` is not reduced but simply removed.

A random-effect term of the form (0 + f1 | gr) or (-1 + f1 | gr) is reduced (if reduce.terms = TRUE) to (1 | gr).

In this exposition it is immaterial whether f1 and f2 are factors or continuous variables.

A random-effect term of the form (1 | gr1/gr2) is automatically expanded to two terms: (1 | gr2:gr1) and (1 | gr1) using findbars_x.

If the model contains structured covariance matrices (introduced with **lme4** version 2.0-0, cf. help(Covariance-class)) other than us (eg. terms such as diag(0 + gr1 | gr2), cs(gr1 | gr2) etc.) ranova behaves as if reduce.terms = FALSE, ie. terms are removed rather than reduced.

Value

an ANOVA-like table with single term deletions of random-effects inheriting from class anova and data.frame with the columns:

npar	number of model parameters.
logLik	the log-likelihood for the model. Note that this is the REML-logLik if the model is fitted with REML.
AIC	the AIC for the model evaluated as -2*(logLik - npar). Smaller is better.
LRT	the likelihood ratio test statistic; twice the difference in log-likelihood, which is asymptotically chi-square distributed.
Df	degrees of freedom for the likelihood ratio test: the difference in number of model parameters.
Pr(>Chisq)	the p-value.

Warning

In certain cases tests of non-nested models may be generated. An example is when (0 + poly(x, 2) | gr) is reduced (the default) to (1 | gr). To our best knowledge non-nested model comparisons are only generated in cases which are statistical nonsense anyway (such as in this example where the random intercept is suppressed).

Note

Note that anova can be used to compare two models and will often be able to produce the same tests as ranova. This is, however, not always the case as illustrated in the examples.

Author(s)

Rune Haubo B. Christensen and Alexandra Kuznetsova

See Also

drop1 for tests of marginal fixed-effect terms and anova for usual anova tables for fixed-effect terms.

Examples

```
# Test reduction of (Days | Subject) to (1 | Subject):
fm1 <- lmer(Reaction ~ Days + (Days|Subject), sleepstudy)
ranova(fm1) # 2 df test

# This test can also be achieved with anova():
fm2 <- lmer(Reaction ~ Days + (1|Subject), sleepstudy)
anova(fm1, fm2, refit=FALSE)

# Illustrate reduce.test argument:
# Test removal of (Days | Subject):
ranova(fm1, reduce.terms = FALSE) # 3 df test

# The likelihood ratio test statistic is in this case:
fm3 <- lm(Reaction ~ Days, sleepstudy)
2*c(logLik(fm1, REML=TRUE) - logLik(fm3, REML=TRUE)) # LRT

# anova() is not always able to perform the same tests as ranova(),
# for example:
anova(fm1, fm3, refit=FALSE) # compares REML with ML and should not be used
anova(fm1, fm3, refit=TRUE) # is a test of ML fits and not what we seek

# Also note that the lmer-fit needs to come first - not an lm-fit:
# anova(fm3, fm1) # does not work and gives an error

# ranova() may not generate all relevant test:
# For the following model ranova() indicates that we should not reduce
# (TVset | Assessor):
fm <- lmer(Coloursaturation ~ TVset * Picture + (TVset | Assessor), data=TVbo)
ranova(fm)
# However, a more appropriate model is:
fm2 <- lmer(Coloursaturation ~ TVset * Picture + (1 | TVset:Assessor), data=TVbo)
anova(fm, fm2, refit=FALSE)
# fm and fm2 has essentially the same fit to data but fm uses 5 parameters
# more than fm2.
```

---

show_tests.anova *Show Hypothesis Tests in ANOVA Tables*

Description

Extracts hypothesis matrices for terms in ANOVA tables detailing exactly which functions of the parameters are being tested in anova tables.

Usage

```
## S3 method for class 'anova'
show_tests(object, fractions = FALSE, names = TRUE, ...)
```

Arguments

`object`	an anova table with a "hypotheses" attribute.
`fractions`	display entries in the hypothesis matrices as fractions?
`names`	if `FALSE` column and row names of the hypothesis matrices are suppressed.
`...`	currently not used.

Value

a list of hypothesis matrices.

Author(s)

Rune Haubo B. Christensen

See Also

`show_tests` for `ls_means` objects.

Examples

```
# Fit basic model to the 'cake' data:
data("cake", package="lme4")
fm1 <- lmer(angle ~ recipe * temp + (1|recipe:replicate), cake)

# Type 3 anova table:
(an <- anova(fm1, type="3"))

# Display tests/hypotheses for type 1, 2, and 3 ANOVA tables:
# (and illustrate effects of 'fractions' and 'names' arguments)
show_tests(anova(fm1, type="1"))
show_tests(anova(fm1, type="2"), fractions=TRUE, names=FALSE)
show_tests(an, fractions=TRUE)
```

`show_tests.ls_means` *Show LS-means Hypothesis Tests and Contrasts*

Description

Extracts the contrasts which defines the LS-mean hypothesis tests.

Usage

```
## S3 method for class 'ls_means'
show_tests(object, fractions = FALSE, names = TRUE, ...)
```

Arguments

`object`	an `ls_means` object.
`fractions`	display contrasts as fractions rather than decimal numbers?
`names`	include row and column names of the contrasts matrices?
`...`	currently not used.

Value

a list of contrast matrices; one matrix for each model term.

Author(s)

Rune Haubo B. Christensen

See Also

`ls_means` for computation of LS-means and `show_tests` for anova objects.

Examples

```
data("cake", package="lme4")
model <- lmer(angle ~ recipe * temp + (1|recipe:replicate), cake)

# LS-means:
(lsm <- ls_means(model))

# Contrasts for LS-means estimates and hypothesis tests:
show_tests(lsm)
```

`step.lmerModLmerTest` *Backward Elimination for Linear Mixed Models*

Description

Backward elimination of random-effect terms followed by backward elimination of fixed-effect terms in linear mixed models.

Usage

```
## S3 method for class 'lmerModLmerTest'
step(
  object,
  ddf = c("Satterthwaite", "Kenward-Roger"),
  alpha.random = 0.1,
  alpha.fixed = 0.05,
  reduce.fixed = TRUE,
```

```
    reduce.random = TRUE,
    keep,
    ...
  )

  ## S3 method for class 'step_list'
  get_model(x, ...)
```

Arguments

object	a fitted model object. For the lmerModLmerTest method an lmer model fit (of class "lmerModLmerTest".)
ddf	the method for computing the denominator degrees of freedom and F-statistics. ddf="Satterthwaite" (default) uses Satterthwaite's method; ddf="Kenward-Roger" uses Kenward-Roger's method.
alpha.random	alpha for random effects elimination
alpha.fixed	alpha for fixed effects elimination
reduce.fixed	reduce fixed effect structure? TRUE by default.
reduce.random	reduce random effect structure? TRUE by default.
keep	an optional character vector of fixed effect terms which should not be considered for eliminated. Valid terms are given by attr(terms(object), "term.labels"). Terms that are marginal to terms in keep will also not be considered for eliminations.
...	currently not used.
x	a step object.

Details

Tests of random-effects are performed using ranova (using reduce.terms = TRUE) and tests of fixed-effects are performed using drop1.

The step method for lmer fits has a print method.

Value

step returns a list with elements "random" and "fixed" each containing anova-like elimination tables. The "fixed" table is based on drop1 and the "random" table is based on ranova (a drop1-like table for random effects). Both tables have a column "Eliminated" indicating the order in which terms are eliminated from the model with zero (0) indicating that the term is not eliminated from the model.

The step object also contains the final model as an attribute which is extractable with get_model(<step_object>).

Author(s)

Rune Haubo B. Christensen and Alexandra Kuznetsova

See Also

drop1 for tests of marginal fixed-effect terms and ranova for a drop1-like table of reduction of random-effect terms.

Examples

```
# Fit a model to the ham dataset:
fm <- lmer(Informed.liking ~ Product*Information+
             (1|Consumer) + (1|Product:Consumer)
           + (1|Information:Consumer), data=ham)

# Backward elimination using terms with default alpha-levels:
(step_res <- step(fm))
final <- get_model(step_res)
anova(final)

## Not run:
# Fit 'big' model:
fm <- lmer(Informed.liking ~ Product*Information*Gender*Age +
             + (1|Consumer) + (1|Consumer:Product) +
             (1|Consumer:Information), data=ham)
step_fm <- step(fm)
step_fm # Display elimination results
final_fm <- get_model(step_fm)

## End(Not run)
```

---

summary.lmerModLmerTest

Summary Method for Linear Mixed Models

Description

Summaries of Linear Mixed Models with coefficient tables including t-tests and p-values using Satterthwaites's or Kenward-Roger's methods for degrees-of-freedom and t-statistics.

Usage

```
## S3 method for class 'lmerModLmerTest'
summary(object, ..., ddf = c("Satterthwaite", "Kenward-Roger", "lme4"))
```

Arguments

object	an lmerModLmerTest object.
...	additional arguments passed on to `lme4::summary.merMod`

ddf
: the method for computing the degrees of freedom and t-statistics. `ddf="Satterthwaite"` (default) uses Satterthwaite's method; `ddf="Kenward-Roger"` uses Kenward-Roger's method, `ddf = "lme4"` returns the lme4-summary i.e., using the summary method for `lmerMod` objects as defined in the **lme4**-package and ignores the `type` argument. Partial matching is allowed.

Details

The returned object is of class `c("summary.lmerModLmerTest", "summary.merMod")` utilizing `print`, `coef` and other methods defined for `summary.merMod` objects. The `"Kenward-Roger"` method use methods from the **pbkrtest** package internally to compute t-statistics and associated degrees-of-freedom.

Value

A summary object with a coefficient table (a `matrix`) including t-values and p-values. The coefficient table can be extracted with `coef(summary(<my-model>))`.

Author(s)

Rune Haubo B. Christensen and Alexandra Kuznetsova

See Also

`contest1D` for one degree-of-freedom contrast tests and `KRmodcomp` for Kenward-Roger F-tests.

Examples

```
# Fit example model:
data("sleepstudy", package="lme4")
fm <- lmer(Reaction ~ Days + (1|Subject) + (0+Days|Subject), sleepstudy)

# Get model summary:
summary(fm) # Satterthwaite df and t-tests

# Extract coefficient table:
coef(summary(fm))

# Use the Kenward-Roger method
if(requireNamespace("pbkrtest", quietly = TRUE))
  summary(fm, ddf="Kenward-Roger")

# The lme4-summary table:
summary(fm, ddf="lme4") # same as summary(as(fm, "lmerMod"))
```

TVbo *Sensory Assesment of B&O TVs*

Description

The TVbo dataset has kindly been made available by the Danish high-end consumer electronics company Bang & Olufsen. The main purpose was to assess 12 different TV sets (products) specified by the two attributes Picture and TVset. 15 different response variables (characteristics of the product) were assessed by a trained panel with 8 assessors.

Usage

```
data(TVbo)
```

Format

Assessor factor with 8 levels assessors.

TVset product factor with 3 levels.

Picture product factor with 4 levels.

In addition the following 15 numeric (response) variables are the characteristics on which the TV sets (products) are assessed:

Coloursaturation, Colourbalance, Noise, Depth, Sharpness, Lightlevel, Contrast, Sharpnessofmovement, Flickeringstationary, Flickeringmovement, Distortion, Dimglasseffect, Cutting, Flossyedges, Elasticeffect.

Examples

```
fm <- lmer(Coloursaturation ~ TVset + Picture + (1|Assessor:TVset) +
             (1|Assessor), data=TVbo)
ranova(fm)
anova(fm)
```

Index