Package: LMMstar (via r-universe)

October 20, 2024

Type Package

Title Repeated Measurement Models for Discrete Times

Version 1.2.0

Date 2024-09-30

Description Companion R package for the course ``Statistical analysis of correlated and repeated measurements for health science researchers" taught by the section of Biostatistics of the University of Copenhagen. It implements linear mixed models where the model for the variance-covariance of the residuals is specified via patterns (compound symmetry, toeplitz, unstructured, ...). Statistical inference for mean, variance, and correlation parameters is performed based on the observed information and a Satterthwaite approximation of the degrees of freedom. Normalized residuals are provided to assess model misspecification. Statistical inference can be performed for arbitrary linear or non-linear combination(s) of model coefficients. Predictions can be computed conditional to covariates only or also to outcome values.

License GPL-3

Encoding UTF-8

URL https://github.com/bozenne/LMMstar

BugReports https://github.com/bozenne/LMMstar/issues

Depends R (>= 3.5.0)

Imports copula, doSNOW, foreach, ggplot2, grid, lava, Matrix, multcomp, nlme, numDeriv, parallel, rlang,

Suggests asht, data.table, ggh4x, ggpubr, lattice, mvtnorm, lme4, lmerTest, mice, nlmeU, optimx, pbapply, psych, Publish, qqtest, R.rsp, rmcorr, scales, testthat

VignetteBuilder R.rsp

RoxygenNote 7.3.1

Collate '0-onload.R' 'BuyseTest_discreteRoot.R' 'LMMstar-package.R' 'LMMstar.options.R' 'add.R' 'anova.R' 'as.array.R' 'as.matrix.R' 'autoplot.R' 'backtransform.R' 'baselineAdjustment.R' 'coef.R' 'collapse.R' 'confint.R' 'constrain.R' 'contrast.R' 'correlate.R' 'df.R' 'doc-data.R' 'dummy.coef.R' 'effects.R' 'estimate.R' 'findPatterns.R' 'fitted.R' 'formula.R' 'iid.R' 'influence.R' 'information.R' 'levels.R' 'lmm.R' 'lmmCC.R' 'logLik.R' 'mlmm.R' 'model.frame.R' 'model.matrix.R' 'model.tables.R' 'moments.R' 'mt.test.R' 'multcomp.R' 'namePatterns.R' 'nesting.R' 'nobs.R' 'operator.R' 'partialCor.R' 'plot.R' 'pool.R' 'precompute.R' 'predict.R' 'print.R' 'profile.R' 'ranef.R' 'rbind.R' 'reformat.R' 'remove.R' 'reparametrize.R' 'repetition.R' 'resample.R' 'residuals.R' 'restaureNA.R' 'sampleRem.R' 'scatterplot.R' 'score.R' 'sigma.R' 'structure-calc_Omega.R' 'structure-calc_d2Omega.R' 'structure-calc_dOmega.R' 'structure-initialization.R' 'structure-skeleton.R' 'structure-skeletonK.R' 'structure-skeletonRho.R' 'structure-skeletonSigma.R' 'structure-update.R' 'structure.R' 'summarize.R' 'summarizeNA.R' 'summary.R' 'terms.R' 'utils.R' 'variable.names.R' 'vcov.R' 'weights.R'

Repository https://bozenne.r-universe.dev

RemoteUrl https://github.com/bozenne/lmmstar

RemoteRef HEAD

RemoteSha fa9178968530cbb66157babf12efc8d734126586

Contents

abetaL *Data From The Bland Altman Study (Long Format)*

Description

Extract data from a longitudinal case control study including 87 patients newly diagnosed with bipolar disorder and 44 age and sex matched healthy controls. Contains demographic data and lifestyle factors at baseline, as well as measures of psychosocial functioning at baseline and 1 year follow-up. This dataset is in the long format (i.e. one line per measurement).

- `id`: study participant.
- `sex`: male (M) or female (F).
- `age`: age in years.
- `group`: bipolar disorder (BD) or healthy control (HC).
- `episode`: whether the patient experience an affective episode during follow-up.
- `visit`: index of time at which pss, fast, and qol measurements where performed.
- `year`: time at which pss, fast, and qol measurements where performed.
- `pss`: perceived stress score.
- `fast`: functioning assessment short test.
- `qol`: WHO quality of life score.
- `educationyears`: years of education including basic school.
- `alcohol`: daily alcohol consumption.
- `missingreason`: reason of drop out or missed visit.

Usage

```
data(abetaL)
```

References

Pech, Josefine, et al. The impact of a new affective episode on psychosocial functioning, quality of life and perceived stress in newly diagnosed patients with bipolar disorder: A prospective one-year case-control study.Journal of Affective Disorders 277 (2020): 486-494.

abetaW *Data From The abeta Study (Wide Format)*

Description

Extract data from a longitudinal case control study including 87 patients newly diagnosed with bipolar disorder and 44 age and sex matched healthy controls. Contains demographic data and lifestyle factors at baseline, as well as measures of psychosocial functioning at baseline and 1 year follow-up. This dataset is in the wide format (i.e. one line per participant).

- `id`: study participant.
- `sex`: male (M) or female (F).
- `age`: age in years.
- `group`: bipolar disorder (BD) or healthy control (HC).
- `episode`: whether the patient experience an affective episode during follow-up.
- `fast0`,`fast1`: functioning assessment short test at baseline and follow-up.
- `qol0`,`qol1`: WHO quality of life score at baseline and follow-up.
- `pss0`,`pss1`: perceived stress score at baseline and follow-up.
- `educationyears`: years of education including basic school.
- `alcohol`: daily alcohol consumption.
- `missingreason`: reason of drop out or missed visit.

Usage

```
data(abetaW)
```

References

Pech, Josefine, et al. "The impact of a new affective episode on psychosocial functioning, quality of life and perceived stress in newly diagnosed patients with bipolar disorder: A prospective one-year case-control study."Journal of Affective Disorders 277 (2020): 486-494.

add *Add Columns to Output*

Description

Auxiliary function that can be used when specifying the argument `columns` (e.g. calling `confint.lmm`) to add columns.

Usage

```
add(...)
```

Arguments

`...` [character vector] name of the columns to be added to the default output.

Value

A character vector

Examples

```
set.seed(10)
dW <- sampleRem(25, n.times = 1, format = "long")
e.lmm <- lmm(Y~X1, data = dW)

confint(e.lmm, columns = add("statistic"))
```

`anova.lmm` *Multivariate Tests For a Linear Mixed Model*

Description

Test linear combinations of parameters from a linear mixed model using Wald test or Likelihood Ratio Test (LRT).

Usage

```
## S3 method for class 'lmm'
anova(
  object,
  effects = NULL,
  rhs = NULL,
  robust = NULL,
  df = NULL,
  univariate = TRUE,
  multivariate = TRUE,
  transform.sigma = NULL,
  transform.k = NULL,
  transform.rho = NULL,
  simplify = NULL,
  ...
)
```

Arguments

`object` a `lmm` object. Only relevant for the anova function.

`effects` [character or numeric matrix] Should the Wald test be computed for all variables (`"all"`), or only variables relative to the mean (`"mean"` or `"fixed"`), or only variables relative to the variance structure (`"variance"`), or only variables relative to the correlation structure (`"correlation"`). Can also be use to specify linear combinations of coefficients or a contrast matrix, similarly to the `linfct` argument of the `multcomp::glht` function.

`rhs` [numeric vector] the right hand side of the hypothesis. Only used when the argument `effects` is a matrix.

`robust` [logical] Should robust standard errors (aka sandwich estimator) be output instead of the model-based standard errors. Can also be `2` compute the degrees-of-freedom w.r.t. robust standard errors instead of w.r.t. model-based standard errors.

`df` [logical] Should degrees-of-freedom be estimated using a Satterthwaite approximation? If yes F-distribution (multivariate) and Student's t-distribution (univariate) are used. Other chi-squared distribution and normal distribution are used.

`univariate` [logical] Should an estimate, standard error, confidence interval, and p-value be output for each hypothesis?

`multivariate` [logical] Should all hypotheses be simultaneously tested using a multivariate Wald test?

`transform.sigma`, `transform.k`, `transform.rho`
are passed to the `vcov` method. See details section in `coef.lmm`.

`simplify` [logical] should only the estimates, variance-covariance with its gradient, and degrees-of-freedom relative to the parameters involved in the Wald test be stored (TRUE) or relative to all model parameters (0.5) along with their iid decomposition (FALSE).

`...` Not used. For compatibility with the generic method.

Details

By default adjustment of confidence intervals and p-values for multiple comparisons is based on the distribution of the maximum-statistic. This is refered to as a single-step Dunnett multiple testing procedures in table II of Dmitrienko et al. (2013). It is performed using the multcomp package with the option `test = adjusted("single-step")` with equal degrees-of-freedom or by simulation using a Student's t copula with unequal degrees-of-freedom (more in the note of the details section of `confint.Wald_lmm`).

Value

A data.frame (LRT) or a list of containing the following elements (Wald):

- `multivariate`: data.frame containing the multivariate Wald test. The column `df.num` refers to the degrees-of-freedom for the numerator (i.e. number of hypotheses) wherease the column `df.denum` refers to the degrees-of-freedom for the denominator (i.e. Satterthwaite approximation).
- `univariate`: data.frame containing each univariate Wald test.
- `glht`: used internally to call functions from the multcomp package.
- `object`: list containing key information about the linear mixed model.
- `args`: list containing argument values from the function call.

References

Dmitrienko, A. and D'Agostino, R., Sr (2013), Traditional multiplicity adjustment methods in clinical trials. Statist. Med., 32: 5172-5218. https://doi.org/10.1002/sim.5990.

See Also

summary.Wald_lmm or confint.Wald_lmm for a summary of the results.
autoplot.Wald_lmm for a graphical display of the results.
rbind.Wald_lmm for combining result across models and adjust for multiple comparisons.

Examples

```
#### simulate data in the long format ####
set.seed(10)
dL <- sampleRem(100, n.times = 3, format = "long")

#### fit Linear Mixed Model ####
eUN.lmm <- lmm(Y ~ visit + X1 + X2 + X5,
               repetition = ~visit|id, structure = "UN", data = dL)

#### Multivariate Wald test ####
## F-tests
anova(eUN.lmm)
anova(eUN.lmm, effects = "all")
anova(eUN.lmm, robust = TRUE, df = FALSE)
summary(anova(eUN.lmm), method = "bonferroni")

## user defined F-test
summary(anova(eUN.lmm, effects = c("X1=0","X2+X5=10")))

## chi2-tests
anova(eUN.lmm, df = FALSE)

## with standard contrast
if(require(multcomp)){
amod <- lmm(breaks ~ tension, data = warpbreaks)
e.amod <- anova(amod, effect = mcp(tension = "Tukey"))
summary(e.amod)
}

#### Likelihood ratio test ####
eUN0.lmm <- lmm(Y ~ X1 + X2, repetition = ~visit|id, structure = "UN", data = dL)
anova(eUN.lmm, eUN0.lmm)

eCS.lmm <- lmm(Y ~ X1 + X2 + X5, repetition = ~visit|id, structure = "CS", data = dL)
anova(eUN.lmm, eCS.lmm)
```

approxAUC *AUC: numerical integral of a function*

Description

Evaluate the Area Under the Curve (AUC) based on discrete observations y = f(x), using interpolation of order 0 (step), 1 (trapezoidal), or 3 (natural cubic splines). The AUC is the integral of a function over an interval [from,to].

Usage

```
approxAUC(
  x,
  y,
  from,
  to,
  method = "trapezoid",
  subdivisions = 100,
  name = "AUC",
  na.rm = FALSE
)
```

Arguments

`x`	[numeric vector] x-values.
`y`	[numeric vector] image of the x-values by the function.
`from`	[numeric] lower border of the intergration domain
`to`	[numeric] upper border of the intergration domain
`method`	[character] the type of interpolation: `"trapezoid"`, `"step"` or `"spline"`.
`subdivisions`	[integer] number of subdivisions to be used when performing numerical integration of the spline. Only relevant when `method="spline"`.
`name`	[character] how to name the output. Can be set to `NULL` or `FALSE` to output a numeric without name.
`na.rm`	[logical] in presence of missing values, should complete.cases of `x` and `y` will be used?

Details

This function is a simplified version of the `AUC` function from the DescTools package.

Value

a numeric value.

Examples

```
## same examples as DescTools::AUC
approxAUC(x=c(1,3), y=c(1,1), from = 1, to = 3)

approxAUC(x=1:3, y=c(1,2,4), from = 1, to = 3)
approxAUC(x=1:3, y=c(1,2,4), from = 1, to = 3, method = "step")

x <- seq(0, pi, length.out=200)
approxAUC(x=x, y=sin(x), from = 0, to = pi)
approxAUC(x=x, y=sin(x), from = 0, to = pi, method = "spline")
```

`autoplot.correlate` *Graphical Display For Correlation Matrix*

Description

Graphical representation for correlation matrix

Usage

```
## S3 method for class 'correlate'
autoplot(
  object,
  index,
  size.text = 16,
  name.legend = "Correlation",
  title = NULL,
  scale = ggplot2::scale_fill_gradient2,
  type.cor = "both",
  low = "blue",
  high = "red",
  mid = "white",
  midpoint = mean(limits),
  limits = c(-1, 1),
  ...
)

## S3 method for class 'correlate'
plot(x, ...)
```

Arguments

`object` [correlate] list of list of correlation matrix

`index` [character vector] optional vector used to select some of the correlation matrix.

`size.text` [numeric, >0] size of the font used to display text.

`name.legend` [character] title for the color scale.

`title` [character] title for the graph.

`scale` [function] color scale used for the correlation.

`type.cor` [character] should the whole correlation matrix be displayed (`"both"`), or only the element in the lower or upper triangle (`"lower"`, `"upper"`).

`low` [character] color used to represent low correlation.

`high` [character] color used to represent high correlation.

`mid` [character] color used to represent moderate correlation.

`midpoint` [numeric] value defining a modereate correlation.

`limits` [numeric vector of length 2] values defining a low and high correlation.

Functions

- `plot(correlate)`: Graphical Display For Correlation Matrix

autoplot.lmm *Graphical Display For a Linear Mixed Model*

Description

Display fitted values or residual plot for the mean, variance, and correlation structure. Can also display quantile-quantile plot relative to the normal distribution.

Usage

```
## S3 method for class 'lmm'
autoplot(
  object,
  type = "fit",
  type.residual = NULL,
  obs.alpha = 0,
  obs.size = NULL,
  facet = NULL,
  facet_nrow = NULL,
  facet_ncol = NULL,
  scales = "fixed",
  labeller = "label_value",
  at = NULL,
  time.var = NULL,
  color = NULL,
  position = NULL,
  ci = TRUE,
  ci.alpha = NULL,
  ylim = NULL,
  mean.size = c(3, 1),
  size.text = 16,
  position.errorbar = "identity",
  ...
)

## S3 method for class 'lmm'
plot(x, ...)
```

Arguments

`object, x` a `lmm` object.

`type` [character] the type of plot

- `"fit"`: fitted values over repetitions.
- `"qqplot"`: quantile quantile plot of the normalized residuals
- `"covariance"`: model (when `type.residual=NULL`) or residual correlation over repetitions
- `"correlation"`: model (when `type.residual=NULL`) or residual correlation over repetitions
- `"scatterplot"`: normalized residuals vs. fitted values (diagnostic for missing non-linear effects),
- `"scatterplot2"`: square root of the normalized residuals vs. fitted values (diagnostic for heteroschedasticity),
- `"partial"`: partial residual plot.

`type.residual` [character] the type of residual to be used or, when `type="partial"`, variable relative to which the partial residuals are computed (argument `variable` in `residuals.lmm`). Not relevant for `type="fit"`.

`obs.alpha` [numeric, 0-1] When not NA, transparency parameter used to display the original data by cluster.

`obs.size` [numeric vector of length 2] size of the point and line for the original data.

`facet` [formula] split the plot into a matrix of panels defined by the variables in the formula. Internally it calls `ggplot2::facet_wrap` or `ggplot2::facet_grid` depending on whether the formula contains a variable on the left hand side.

`facet_nrow` [integer] number of rows of panels in the graphical display.

`facet_ncol` [integer] number of columns of panels in the graphical display.

`scales, labeller`
[character] Passed to `ggplot2::facet_wrap`.

`at` [data.frame] values for the covariates at which to evaluate the fitted values or partial residuals.

`time.var` [character] x-axis variable for the plot.

`color` [character] name of the variable in the dataset used to color the curve. No color is used when set to `FALSE`.

`position` [character] relative position of the points when colored according to a variable.

`ci` [logical] should confidence intervals be displayed?

`ci.alpha` [numeric, 0-1] When not NA, transparency parameter used to display the confidence intervals.

`ylim` [numeric vector of length 2] the lower and higher value of the vertical axis, or the range of the color scale for correlation or covariance plot.

`mean.size` [numeric vector of length 2] size of the point and line for the mean trajectory.

`size.text` [numeric, >0] size of the font used to display text.

`position.errorbar`
[character] relative position of the errorbars.

`...` arguments passed to the `.autofit`, `.ggHeatmap` or `autoplot.residual_lmm` functions.

Value

A list with two elements

- `data`: data used to create the graphical display.
- `plot`: ggplot object.

Functions

- `plot(lmm)`: Graphical Display For a Linear Mixed Model

See Also

`plot.lmm` for other graphical display (residual plots, partial residual plots).

Examples

```
if(require(ggplot2)){

#### simulate data in the long format ####
set.seed(10)
dL <- sampleRem(100, n.times = 3, format = "long")
dL$X1 <- as.factor(dL$X1)

#### fit Linear Mixed Model ####
eCS.lmm <- lmm(Y ~ visit + X1 + X6,
               repetition = ~visit|id, structure = "CS", data = dL, df = FALSE)

#### model fit ####
plot(eCS.lmm, type = "fit", facet =~X1)
## customize display
gg <- autoplot(eCS.lmm, type = "fit", facet =~X1)$plot
gg + coord_cartesian(ylim = c(0,6))
## restrict to specific covariate value
plot(eCS.lmm, type = "fit", at = data.frame(X6=1), color = "X1")

#### qqplot ####
plot(eCS.lmm, type = "qqplot")
plot(eCS.lmm, type = "qqplot", engine.qqplot = "qqtest")

#### residual correlation ####
plot(eCS.lmm, type = "correlation")

#### residual trend ####
plot(eCS.lmm, type = "scatterplot")

#### residual heteroschedasticity ####
plot(eCS.lmm, type = "scatterplot2")

#### partial residuals ####
plot(eCS.lmm, type = "partial", type.residual = "visit")
plot(eCS.lmm, type = "partial", type.residual = c("(Intercept)","X1","visit"))
plot(eCS.lmm, type = "partial", type.residual = c("(Intercept)","X1","visit"),
facet = ~X1)
}
```

`autoplot.mlmm` *Graphical Display of Wald Tests For Multiple Linear Mixed Models.*

Description

Display estimated linear contrasts applied on parameters from subgroup-specific linear mixed models.

Usage

```
## S3 method for class 'mlmm'
autoplot(object, type = "forest", ...)

## S3 method for class 'mlmm'
plot(x, facet_nrow = NULL, facet_ncol = NULL, labeller = "label_value", ...)
```

Functions

- `plot(mlmm)`: Graphical Display For Multiple Linear Mixed Models

`autoplot.partialCor` *Graphical Display For Partial Correlation*

Description

Extract and display the correlation modeled via the linear mixed model.

Usage

```
## S3 method for class 'partialCor'
autoplot(
  object,
  size.text = 16,
  limits = c(-1, 1.00001),
  low = "blue",
  mid = "white",
  high = "red",
  midpoint = 0,
  ...
)

## S3 method for class 'partialCor'
plot(x, ...)
```

Arguments

`object, x` a `partialCor` object.

`size.text` [numeric, >0] size of the font used to display text.

`limits` [numeric vector of length 2] minimum and maximum value of the colorscale relative to the correlation.

`low, mid, high` [character] color for the the colorscale relative to the correlation.

`midpoint` [numeric] correlation value associated with the color defined by argument `mid`.

`...` Not used. For compatibility with the generic method.

Value

A list with two elements

- `data`: data used to create the graphical display.
- `plot`: ggplot object.

Functions

- `plot(partialCor)`: Graphical Display For Partial Correlation

Examples

```
if(require(ggplot2)){
data(gastricbypassL, package = "LMMstar")

e.pCor <- partialCor(c(weight,glucagonAUC)~time, repetition = ~visit|id,
                     data = gastricbypassL)
plot(e.pCor)
}
```

autoplot.profile_lmm *Graphical Display of Profile Likelihood*

Description

Graphical representation of the profile likelihood from a linear mixed model

Usage

```
## S3 method for class 'profile_lmm'
autoplot(
  object,
  type = "logLik",
  quadratic = TRUE,
  ci = FALSE,
  size = c(3, 2, 1, 1),
  linetype = c("dashed", "dashed", "dashed"),
  shape = 19,
  scales = "free",
  nrow = NULL,
  ncol = NULL,
  ...
)

## S3 method for class 'profile_lmm'
plot(x, ...)
```

Arguments

object, x	an object of class profile_lmm, output of the profile.lmm function.
type	[character] Should the log-likelihood ("logLik") or the ratio to the maximum likelihood ("ratio") be displayed?
quadratic	[logical] Should a quadratic approximation of the likelihood be displayed?
ci	[logical] Should a 95% confidence intervals obtained from the Wald test (vertical lines) and Likelihood ratio test (horizontal line) be displayed?
size	[numeric vector of length 4] Size of the point for the MLE, width of the line representing the likelihood, width of the corresponding quadratic approximation, and width of the line representing the confidence intervals.
linetype	[integer vector of length 2] type of line used to represent the quadratic approximation of the likelihood and the confidence intervals.
shape	[integer, >0] type of point used to represent the MLE.
scales, nrow, ncol	argument passed to ggplot2::facet_wrap.
...	Not used. For compatibility with the generic method.

Value

A list with three elements

- data.fit: data containing the quadratice approximation of the log-likelihood
- data.ci: data containing the confidence intervals.
- plot: ggplot object.

Functions

- plot(profile_lmm): Display Contour of the log-Likelihood

---

autoplot.residuals_lmm

Graphical Display of the Residuals

Description

Graphical representation of the residuals from a linear mixed model. Require a long format (except for the correlation where both format are accepted) and having exported the dataset along with the residual (argument keep.data when calling residuals.lmm).

Usage

```
## S3 method for class 'residuals_lmm'
autoplot(
  object,
  type = NULL,
  type.residual = NULL,
  time.var = NULL,
  facet = NULL,
  facet_nrow = NULL,
  facet_ncol = NULL,
  scales = "fixed",
  labeller = "label_value",
  engine.qqplot = "ggplot2",
  add.smooth = TRUE,
  digits.cor = 2,
  size.text = 16,
  color = NULL,
  obs.size = NULL,
  mean.size = c(3, 1),
  ci.alpha = 0.25,
  ylim = NULL,
  position = NULL,
  ...
)
```

```
## S3 method for class 'residuals_lmm'
plot(x, ...)
```

Arguments

`object, x` an object of class `residuals_lmm`, output of the `residuals.lmm` function.

`type` [character] Should a qqplot (`"qqplot"`), or a heatmap of the correlation between residuals (`"correlation"`, require wide format), or a plot of residuals along the fitted values (`"scatterplot"`, require long format) be displayed?

`type.residual` [character] Type of residual for which the graphical representation should be made.

`time.var` [character] x-axis variable for the plot. Only relevant when argument type is one of `"scatterplot"`, `"scatterplot2"`, `"partial"`, `"partial-center"`,

`facet` [formula] split the plot into a matrix of panels defined by the variables in the formula. Internally it calls `ggplot2::facet_wrap` or `ggplot2::facet_grid` depending on whether the formula contains a variable on the left hand side.

`facet_nrow` [integer] number of rows of panels in the graphical display.

`facet_ncol` [integer] number of columns of panels in the graphical display.

`scales, labeller` [character] Passed to `ggplot2::facet_wrap`.

`engine.qqplot` [character] Should ggplot2 or qqtest be used to display quantile-quantile plots? Only used when argument `type` is `"qqplot"`.

`add.smooth` [logical] should a local smoother be used to display the mean of the residual values across the fitted values. Only relevant for when argument `type` is `"scatterplot"`.

`digits.cor` [integer, >0] Number of digit used to display the correlation coefficients? No correlation coefficient is displayed when set to 0. Only used when argument `plot` is `"correlation"`.

`size.text` [numeric, >0] Size of the font used to displayed text when using ggplot2.

`color` [character] color of the dots representing the observations. When displaying partial residuals, should contain a second color indicating how to display the model fit.

`obs.size` [numeric vector] size of the dots representing the observations.

`mean.size` [numeric vector of length 2] size of the point and line for the mean trajectory.

`ci.alpha` [numeric, 0-1] When not NA, transparency parameter used to display the confidence intervals.

`ylim` [numeric vector of length 2] the lower and higher value of the vertical axis, or the range of the color scale for correlation or covariance plot.

`position` [character] relative position of the points when colored according to a variable.

`...` Not used. For compatibility with the generic method.

Value

A list with two elements

- `data`: data used to generate the plot.
- `plot`: ggplot object.

Functions

- `plot(residuals_lmm)`: Graphical Display of the Residuals

---

autoplot.summarize *Graphical Display of the Descriptive Statistics*

Description

Graphical representation of the descriptive statistics.

Usage

```
## S3 method for class 'summarize'
autoplot(
  object,
  type = "mean",
  variable = NULL,
  size.text = 16,
  linewidth = 1.25,
  size = 3,
  ...
)

## S3 method for class 'summarize'
plot(x, ...)
```

Arguments

`object, x`	an object of class `summarize`, output of the `summarize` function.
`type`	[character] the summary statistic that should be displayed: `"mean"`, `"sd"`, ...
`variable`	[character] type outcome relative to which the summary statistic should be displayed. Only relevant when multiple variables have been used on the left hand side of the formula when calling `summarize`.
`size.text`	[numeric, >0] size of the text in the legend, x- and y- labels.
`linewidth`	[numeric, >0] thickness of the line connecting the points.
`size`	[numeric, >0] width of the points.
`...`	additional arguments passed to .ggHeatmap when displaying the correlation: • name.time [character] title for the x- and y- axis.

- digits.cor [integer, >0] number of digits used to display the correlation.
- name.legend [character] title for the color scale.
- title [character] title for the graph.
- scale [function] color scale used for the correlation.
- type.cor [character] should the whole correlation matrix be displayed (`"both"`), or only the element in the lower or upper triangle (`"lower"`, `"upper"`).
- low [character] color used to represent low correlation.
- mid [character] color used to represent moderate correlation.
- high [character] color used to represent high correlation.
- midpoint [numeric] value defining a modereate correlation.
- limits [numeric vector of length 2] values defining a low and high correlation.

Value

A list with two elements

- `data`: data used to generate the plot.
- `plot`: ggplot object.

Functions

- `plot(summarize)`: Graphical Display of Missing Data Pattern

Examples

```
data(gastricbypassL, package = "LMMstar")
dtS <- summarize(weight ~ time, data = gastricbypassL)
plot(dtS)
dtS <- summarize(glucagonAUC + weight ~ time|id, data = gastricbypassL, na.rm = TRUE)
plot(dtS, variable = "glucagonAUC")
plot(dtS, variable = "glucagonAUC", type = "correlation", size.text = 1)
```

`autoplot.summarizeNA` *Graphical Display of Missing Data Pattern*

Description

Graphical representation of the possible missing data patterns in the dataset.

Usage

```
## S3 method for class 'summarizeNA'
autoplot(
  object,
  variable = NULL,
  size.text = 16,
  add.missing = " missing",
  order.pattern = NULL,
  decreasing.order = FALSE,
  title = NULL,
  labeller = "label_value",
  ...
)

## S3 method for class 'summarizeNA'
plot(x, ...)
```

Arguments

`object, x` a `summarizeNA` object, output of the `summarizeNA` function.

`variable` [character] variable for which the missing patterns should be displayed. Only required when the argument `repetition` has been specified when calling `summarizeNA`.

`size.text` [numeric, >0] size of the font used to display text.

`add.missing` [logical] should the number of missing values per variable be added to the x-axis tick labels.

`order.pattern` [numeric vector or character] in which order the missing data pattern should be displayed. Can either be a numeric vector indexing the patterns or a character refering to order the patterns per number of missing values (`"n.missing"`) or number of observations (`"frequency"`).

`title` [character] title of the graphical display. Passed to `ggplot2::ggtitle`.

`labeller` [character] how to label each facet: only the value (`"label_value"`) or with also the variable name (`"label_both"`). Passed to `ggplot2::facet_wrap`.

`...` Not used. For compatibility with the generic method.

`decreasing` [logical] when ordering the missing data pattern (see argument `order.pattern`) should the sort order be decreasing? Passed to `base::order`.

Value

A list with two elements

- `data`: data used to create the graphical display.
- `plot`: ggplot object.

Functions

- `plot(summarizeNA)`: Graphical Display of Missing Data Pattern

`autoplot.Wald_lmm` *Graphical Display For Wald Tests Applied to a Linear Mixed Model*

Description

Display estimated linear contrast applied on parameters from a linear mixed model.

Usage

```
## S3 method for class 'Wald_lmm'
autoplot(object, type = "forest", size.text = 16, add.args = NULL, ...)

## S3 method for class 'Wald_lmm'
plot(x, ...)
```

Arguments

`object, x`	a `Wald_lmm` object.
`type`	[character] what to display: a forest plot (`"forest"`) or a heatmap (`"heat"`).
`size.text`	[numeric, >0] size of the font used to display text.
`add.args`	[list] additional arguments used to customized the graphical display. Must be a named list. See details.
`...`	arguments passed to the confint method.

Details

Argument **add.args**: parameters specific to the forest plot:

- `color`: [logical] should the estimates be colored by global null hypothesis, e.g. when testing the effect of a 3 factor covariate, the two corresponding coefficient will have the same color. Alternatively a vector of positive integers giving the color with which each estimator should be displayed.
- `color`: [logical] should the estimates be represented by a different shape per global null hypothesis, e.g. when testing the effect of a 3 factor covariate, the two corresponding coefficient will have the same type of point. Alternatively a vector of positive integers describing the shape to be used for each estimator.
- `ci`: [logical] should confidence intervals be displayed?
- `size.estimate`: [numeric, >0] size of the dot used to display the estimates.
- `size.ci`: [numeric, >0] thickness of the line used to display the confidence intervals.
- `width.ci`: [numeric, >0] width of the line used to display the confidence intervals.
- `size.null`: [numeric, >0] thickness of the line used to display the null hypothesis.

Parameters specific to the heatmap plot:

- `limits`: [numeric vector of length 2] minimum and maximum value of the colorscale relative to the correlation.

- low, mid, high: [character] color for the the colorscale relative to the correlation.
- midpoint: [numeric] correlation value associated with the color defined by argument mid

Value

A list with two elements

- data: data used to create the graphical display.
- plot: ggplot object.

Functions

- plot(Wald_lmm): Graphical Display For Wald Tests Applied to a Linear Mixed Model

Examples

```
## From the multcomp package
if(require(datasets) && require(ggplot2)){

## only tests with 1 df
ff <- Fertility ~ Agriculture + Examination + Education + Catholic + Infant.Mortality
e.lmm <- lmm(ff, data = swiss)
e.aovlmm <- anova(e.lmm)

autoplot(e.aovlmm, type = "forest")
autoplot(e.aovlmm, type = "heat") ## 3 color gradient
autoplot(e.aovlmm, type = "heat", add.args = list(mid = NULL)) ## 2 color gradient

## test with more than 1 df
e.lmm2 <- lmm(breaks ~ tension + wool, data = warpbreaks)
e.aovlmm2 <- anova(e.lmm2)
autoplot(e.aovlmm2)
autoplot(e.aovlmm2, add.args = list(color = FALSE, shape = FALSE))
}
```

baselineAdjustment *Perform Baseline Adjustment*

Description

Create a new variable based on a time variable and a group variable where groups are constrained to be equal at specific timepoints.

Usage

```
baselineAdjustment(
  object,
  variable,
  repetition,
  constrain,
  new.level = NULL,
  collapse.time = NULL
)
```

Arguments

`object` [data.frame] dataset

`variable` [character] Column in the dataset to be constrained at specific timepoints.

`repetition` [formula] Time and cluster structure, typically `~time|id`. See examples below.

`constrain` [vector] Levels of the time variable at which the variable is constained.

`new.level` [character or numeric] Level used at the constraint. If `NULL`, then the first level of the variable argument is used.

`collapse.time` [character] When not `NULL` character used to combine the time and argument variable into a new (interaction) variable.

Value

A vector of length the number of rows of the dataset.

Examples

```
data(ncgsL, package = "LMMstar")
ncgsL$group <- relevel(ncgsL$group, "placebo")

## baseline adjustment 1
ncgsL$treat <- baselineAdjustment(ncgsL, variable = "group",
               repetition= ~ visit|id, constrain = 1)
table(treat = ncgsL$treat, visit = ncgsL$visit, group = ncgsL$group)

ncgsL$treattime <- baselineAdjustment(ncgsL, variable = "group",
                   repetition= ~ visit|id, constrain = 1, collapse.time = ".")
table(treattime = ncgsL$treattime, visit = ncgsL$visit, group = ncgsL$group)

## baseline adjustment 2
ncgsL$treat2 <- baselineAdjustment(ncgsL, variable = "group",
                 new.level = "baseline",
                 repetition= ~ visit|id, constrain = 1)
table(treat = ncgsL$treat2, visit = ncgsL$visit, group = ncgsL$group)

ncgsL$treattime2 <- baselineAdjustment(ncgsL, variable = "group",
                   new.level = "baseline",
                   repetition= ~ visit|id, constrain = 1, collapse.time = ".")
table(treattime = ncgsL$treattime2, visit = ncgsL$visit, group = ncgsL$group)
```

`blandAltmanL` *Data From The Bland Altman Study (Long Format)*

Description

Data From The Bland Altman Study where two methods to measure the peak expiratory flow rate (PEFR) where compared. This dataset is in the long format (i.e. one line per measurement).

- `id`: patient identifier.
- `replicate`: index of the measurement (first or second).
- `method`: device used to make the measurement (Wright peak flow meter or mini Wright peak flow meter).
- `pefr`: measurement (peak expiratory flow rate).

Usage

```
data(blandAltmanL)
```

References

Bland & Altman, Statistical methods for assessing agreement between two methods of clinical measurement, Lancet, 1986; i: 307-310.

`blandAltmanW` *Data From The Bland Altman Study (Wide Format)*

Description

Data From The Bland Altman Study where two methods to measure the peak expiratory flow rate (PEFR) where compared. This dataset is in the wide format (i.e. one line per patient).

- `id`: patient identifier.
- `wright1`: first measurement made with a Wright peak flow meter.
- `wright2`: second measurement made with a Wright peak flow meter.
- `mini1`: first measurement made with a mini Wright peak flow meter.
- `mini2`: second measurement made with a mini Wright peak flow meter.

Usage

```
data(blandAltmanW)
```

References

Bland & Altman, Statistical methods for assessing agreement between two methods of clinical measurement, Lancet, 1986; i: 307-310.

bloodpressureL *Data From The Blood Pressure Study (Long Format)*

Description

Data from a cross-over trial comparing the impact of three formulations of a drug on the blood pressure. The study was conducted on 12 male volunteers randomly divided into tree groups and receiving each of the three formulations with a wash-out period of one week.

- `id`: patient identifier.
- `sequence`: sequence of treatment .
- `treatment`: formulation of the treatment A (50 mg tablet) B (100 mg tablet) C (sustained-release formulation capsule)
- `period`: time period (in weeks).
- `duration`: duration of the drug (in hours).

Usage

```
data(bloodpressureL)
```

References

TO ADD

calciumL *Data From The Calcium Supplements Study (Long Format)*

Description

Data from a randomized study including 112 girls at age 11 investigate the effect of a calcium supplement (n=55) vs. placebo (n=57) on bone mineral density over a 2 year follow-up. The clinical question is: does a calcium supplement help to increase bone gain in adolescent women? This dataset is in the long format (i.e. one line per measurement).

- `girl`: patient identifier.
- `grp`: treatment group: calcium supplement (coded `C`) or placebo (coded `P`).
- `visit`: visit index.
- `bmd`: bone mineral density (mg/cm3).
- `time.obs`: visit time (in years).
- `time.num`: scheduled visit time (numeric variable, in years).
- `time.fac`: scheduled visit time (factor variable).

Usage

```
data(calciumL)
```

References

TO ADD

`calciumW` *Data From The Calcium Supplements Study (Wide Format)*

Description

Data from a randomized study including 112 girls at age 11 investigate the effect of a calcium supplement (n=55) vs. placebo (n=57) on bone mineral density over a 2 year follow-up. The clinical question is: does a calcium supplement help to increase bone gain in adolescent women? This dataset is in the wide format (i.e. one line per patient).

- `girl`: patient identifier
- `grp`: treatment group: calcium supplement (coded `C`) or placebo (coded `P`).
- `obstime1`: time after the start of the study at which the first visit took place (in years).
- `obstime2`: time after the start of the study at which the second visit took place (in years).
- `obstime3`: time after the start of the study at which the third visit took place (in years).
- `obstime4`: time after the start of the study at which the fourth visit took place (in years).
- `obstime5`: time after the start of the study at which the fifth visit took place (in years).
- `bmd1`: bone mineral density measured at the first visit (in mg/cm3).
- `bmd2`: bone mineral density measured at the second visit (in mg/cm3).
- `bmd3`: bone mineral density measured at the third visit (in mg/cm3).
- `bmd4`: bone mineral density measured at the fourth visit (in mg/cm3).
- `bmd5`: bone mineral density measured at the fifth visit (in mg/cm3).

Usage

```
data(calciumW)
```

References

Vonesh and Chinchilli 1997. Linear and Nonlinear models for the analysis of repeated measurement (Table 5.4.1 on page 228). New York: Marcel Dekker.

`ckdL` *CKD long*

Description

TODO

- `id`: patient identifier.
- `allocation`:
- `sex`:
- `age`:
- `visit`:
- `time`:
- `pwv`:
- `aix`:
- `dropout`:

Usage

```
data(ckdL)
```

References

TO ADD

`ckdW` *CKD wide*

Description

TODO

- `id`: patient identifier.
- `allocation`:
- `sex`:
- `age`:
- `pwv0`:
- `pwv12`:
- `pwv24`:
- `aix0`:
- `aix12`:
- `aix24`:
- `dropout`:

Usage

```
data(ckdW)
```

References

TO ADD

coef.lmm *Extract Coefficients From a Linear Mixed Model*

Description

Extract estimated parameters from a linear mixed model.

Usage

```
## S3 method for class 'lmm'
coef(
  object,
  effects = NULL,
  p = NULL,
  transform.sigma = NULL,
  transform.k = NULL,
  transform.rho = "none",
  transform.names = TRUE,
  simplify = TRUE,
  ...
)
```

Arguments

`object` a `lmm` object.

`effects` [character] Should all coefficients be output (`"all"`), or only coefficients relative to the mean (`"mean"` or `"fixed"`), or only coefficients relative to the variance structure (`"variance"`), or only coefficients relative to the correlation structure (`"correlation"`). or the random effects (`"ranef"`) when using a random effect model.

`p` [numeric vector] value of the model coefficients to be used. Only relevant if differs from the fitted values.

`transform.sigma` [character] Transformation used on the variance coefficient for the reference level. One of `"none"`, `"log"`, `"square"`, `"logsquare"` - see detail##' @param transform.k [character] Transformation used on the variance coefficients relative to the other levels. One of `"none"`, `"log"`, `"square"`, `"logsquare"`, `"sd"`, `"logsd"`, `"var"`, `"logvar"` - see details.

`transform.rho`	[character] Transformation used on the correlation coefficients. One of `"none"`, `"atanh"`, `"cov"` - see details.
`transform.names`	[logical] Should the name of the coefficients be updated to reflect the transformation that has been used?
`simplify`	[logical] Omit from the output the attribute containing the type of each parameter (mu/sigma/k/rho) and the corresponding variance parameters (sigma/k.x/k.y).
`...`	Not used. For compatibility with the generic method.

Details

transform.sigma:

- `"none"` ouput residual standard error.
- `"log"` ouput log-transformed residual standard error.
- `"square"` ouput residual variance.
- `"logsquare"` ouput log-transformed residual variance.

transform.k:

- `"none"` ouput ratio between the residual standard error of the current level and the reference level.
- `"log"` ouput log-transformed ratio between the residual standard errors.
- `"square"` ouput ratio between the residual variances.
- `"logsquare"` ouput log-transformed ratio between the residual variances.
- `"sd"` ouput residual standard error of the current level.
- `"logsd"` ouput residual log-transformed standard error of the current level.
- `"var"` ouput residual variance of the current level.
- `"logvar"` ouput residual log-transformed variance of the current level.

transform.rho:

- `"none"` ouput correlation coefficient.
- `"atanh"` ouput correlation coefficient after tangent hyperbolic transformation.
- `"cov"` ouput covariance coefficient.

Value

A vector with the value of the model coefficients.
When using `simplify=FALSE` the character strings in attribute `"type"` refer to:

- `"mu"`: mean parameters.
- `"sigma"`: standard deviation parameters,

- "k": ratio between standard deviation,
- "rho": correlation parameter

The character strings in attribute "sigma" refer, for each parameter, to a possible corresponding standard deviation parameter. Those in attribute "k.x" and "k.y" refer to the ratio parameter. NA indicates no corresponding standard deviation or ratio parameter.

See Also

confint.lmm or model.tables.lmm for a data.frame containing estimates with their uncertainty.

Examples

```
## simulate data in the long format
set.seed(10)
dL <- sampleRem(100, n.times = 3, format = "long")

## fit linear mixed model
eUN.lmm <- lmm(Y ~ X1 + X2 + X5, repetition = ~visit|id, structure = "UN", data = dL, df = FALSE)

## output coefficients
coef(eUN.lmm)
coef(eUN.lmm, effects = "variance", transform.k = "sd")
coef(eUN.lmm, effects = "all", simplify = FALSE)
```

---

coef.mlmm *Extract Coefficients From Multiple Linear Mixed Models*

Description

Combine estimated parameters or linear contrasts applied on parameters from group-specific linear mixed models.

Usage

```
## S3 method for class 'mlmm'
coef(
  object,
  effects = "Wald",
  method = "none",
  p = NULL,
  ordering = "model",
  backtransform = object$args$backtransform,
  transform.sigma = "none",
  transform.k = "none",
  transform.rho = "none",
  transform.names = TRUE,
  simplify = TRUE,
  ...
)
```

Arguments

`object` a `mlmm` object.

`effects` [character] By default will output the estimates relative to the hypotheses being tested (`"Wald"`). But can also output all model coefficients (`"all"`), or only coefficients relative to the mean (`"mean"` or `"fixed"`), or only coefficients relative to the variance structure (`"variance"`), or only coefficients relative to the correlation structure (`"correlation"`).

`method` [character vector] should the estimated value for the linear contrasts be output (one of `"none"`, `"bonferroni"`, ..., `"single-step2"`) and/or pooled linear contrast estimate(s) (`"average"`, `"pool.se"`, `"pool.gls"`, `"pool.gls1"`, `"pool.rubin"`, `"p.rejection"`)? Only relevant when `effects = "Wald"`.

`p` [list of numeric vector] values for the model parameters to be used to evaluate the estimates relative to the hypotheses being tested. Only relevant if differs from the fitted values.

`ordering` [character] should the output be ordered by name of the linear contrast (`"contrast"`) or by model (`"model"`).

`backtransform` [logical] should the estimate be back-transformed? Only relevant when `effects="Wald"`.

`transform.sigma`
[character] Transformation used on the variance coefficient for the reference level. One of `"none"`, `"log"`, `"square"`, `"logsquare"`. Ignored when `effects="Wald"`.

`transform.k` [character] Transformation used on the variance coefficients relative to the other levels. One of `"none"`, `"log"`, `"square"`, `"logsquare"`, `"sd"`, `"logsd"`, `"var"`, `"logvar"`. Ignored when `effects="Wald"`.

`transform.rho` [character] Transformation used on the correlation coefficients. One of `"none"`, `"atanh"`, `"cov"`. Ignored when `effects="Wald"`.

`transform.names`
[logical] Should the name of the coefficients be updated to reflect the transformation that has been used? Ignored when `effects="Wald"`.

`simplify` [logical] should the output be a vector or a list with one element specific to each possible ordering (i.e. contrast or model). Only relevant when argument `method` refers to multiple comparisons and not to a pooling method.

`...` Not used. For compatibility with the generic method.

---

coef.rbindWald_lmm *Extract Coefficients From Combined Wald Tests Applied to Linear Mixed Models*

Description

Combine estimated values across linear contrasts applied on parameters from different linear mixed models.

Usage

```
## S3 method for class 'rbindWald_lmm'
coef(
  object,
  effects = "Wald",
  method = "none",
  ordering = NULL,
  transform.names = TRUE,
  backtransform = NULL,
  simplify = TRUE,
  ...
)
```

Arguments

`object` a `rbindWald_lmm` object.

`effects` [character] should the linear contrasts involved in the Wald test be output (`"Wald"`), or the value of the linear mixed model parameters (`"all"`)?

`method` [character vector] should the estimated value for the linear contrasts be output (one of `"none"`, `"bonferroni"`, ..., `"single-step2"`) and/or pooled linear contrast estimate(s) (`"average"`, `"pool.se"`, `"pool.gls"`, `"pool.gls1"`, `"pool.rubin"`, `"p.rejection"`)? Only relevant when `effects = "Wald"`.

`ordering` [character] should the output be ordered by name of the linear contrast (`"contrast"`) or by model (`"model"`).

`transform.names` [logical] should the name of the coefficients be updated to reflect the transformation that has been used? Only relevant when `effects="all"`.

`backtransform` [logical] should the estimates be back-transformed?

`simplify` [logical] should the output be a vector or a list with one element specific to each possible ordering (i.e. contrast or model). Only relevant when argument `method` refers to multiple comparisons and not to a pooling method.

`...` Not used. For compatibility with the generic method.

Details

Argument **effects**: when evaluating the proportion of rejected hypotheses (`effects="p.rejection"`) a `"single-step"` method will be used by default to evaluate the critical quantile. This can be changed by adding adjustment method, e.g. `effects=c("bonferronin","p.rejection"`, in the argument.

`coef.Wald_lmm` *Extract Coefficients From Wald Tests Applied to a Linear Mixed Model*

Description

Extract estimated value of linear contrasts applied on parameters from a linear mixed model.

Usage

```
## S3 method for class 'Wald_lmm'
coef(
  object,
  effects = "Wald",
  backtransform = NULL,
  transform.names = TRUE,
  simplify = TRUE,
  ...
)
```

Arguments

`object` a `Wald_lmm` object.

`effects` [character] should the linear contrasts involved in the Wald test be output (`"Wald"`), or the value of the linear mixed model parameters (`"all"`)?

`backtransform` [logical] should the estimates be back-transformed?

`transform.names` [logical] Should the name of the coefficients be updated to reflect the transformation that has been used? Only relevant when `effects="all"`.

`simplify` [logical] omit from the output the attribute containing the type of each parameter or contrast (mu/sigma/k/rho).

`...` Not used. For compatibility with the generic method.

`confint.lmm` *Confidence Intervals for a Linear Mixed Model*

Description

Compute confidence intervals (CIs) relative to parameters from a linear mixed model.

Usage

```
## S3 method for class 'lmm'
confint(
  object,
  parm = NULL,
  level = 0.95,
  effects = NULL,
  robust = FALSE,
  null = NULL,
  columns = NULL,
  df = NULL,
  type.information = NULL,
  transform.sigma = NULL,
  transform.k = NULL,
  transform.rho = NULL,
  transform.names = TRUE,
  backtransform = NULL,
  ...
)
```

Arguments

`object` a `lmm` object.

`parm` Not used. For compatibility with the generic method.

`level` [numeric,0-1] the confidence level of the confidence intervals.

`effects` [character] Should the CIs/p-values for all coefficients be output (`"all"`), or only for mean coefficients (`"mean"` or `"fixed"`), or only for variance coefficients (`"variance"`), or only for correlation coefficients (`"correlation"`).

`robust` [logical] Should robust standard errors (aka sandwich estimator) be output instead of the model-based standard errors. Can also be `2` compute the degrees-of-freedom w.r.t. robust standard errors instead of w.r.t. model-based standard errors.

`null` [numeric vector] the value of the null hypothesis relative to each coefficient.

`columns` [character vector] Columns to be output. Can be any of `"estimate"`, `"se"`, `"statistic"`, `"df"`, `"null"`, `"lower"`, `"upper"`, `"p.value"`.

`df` [logical] Should a Student's t-distribution be used to model the distribution of the coefficient. Otherwise a normal distribution is used.

`type.information`, `transform.sigma`, `transform.k`, `transform.rho`, `transform.names`
are passed to the `vcov` method. See details section in `coef.lmm`.

`backtransform` [logical] should the variance/covariance/correlation coefficient be backtransformed?

`...` Not used. For compatibility with the generic method.

Value

A data.frame containing some of the following coefficient (in rows):

- column estimate: the estimate.
- column se: the standard error.
- column statistic: the test statistic.
- column df: the degrees-of-freedom.
- column lower: the lower bound of the confidence interval.
- column upper: the upper bound of the confidence interval.
- column null: the null hypothesis.
- column p.value: the p-value relative to the null hypothesis.

See Also

the function anova to perform inference about linear combinations of coefficients and adjust for multiple comparisons.

coef.lmm for a simpler output (e.g. only estimates).
model.tables.lmm for a more detailed output (e.g. with p-value).

Examples

```
#### simulate data in the long format ####
set.seed(10)
dL <- sampleRem(100, n.times = 3, format = "long")

#### fit Linear Mixed Model ####
eUN.lmm <- lmm(Y ~ X1 + X2 + X5, repetition = ~visit|id, structure = "UN", data = dL)

#### Confidence intervals ####
## based on a Student's t-distribution with transformation
confint(eUN.lmm, effects = "all")
## based on a Student's t-distribution without transformation
confint(eUN.lmm, effects = "all",
        transform.sigma = "none", transform.k = "none", transform.rho = "none")
## based on a Student's t-distribution transformation but not backtransformed
confint(eUN.lmm, effects = "all", backtransform = FALSE)
## based on a Normal distribution with transformation
confint(eUN.lmm, df = FALSE)
```

confint.mlmm *Confidence Intervals for Multiple Linear Mixed Models*

Description

Compute pointwise or simultaneous confidence intervals relative to parameter or linear contrasts of parameters from group-specific linear mixed models. Can also output p-values (corresponding to pointwise confidence intervals) or adjusted p-values (corresponding to simultaneous confidence intervals).

Usage

```
## S3 method for class 'mlmm'
confint(
  object,
  parm = NULL,
  level = 0.95,
  method = NULL,
  df = NULL,
  columns = NULL,
  backtransform = NULL,
  ordering = "parameter",
  ...
)
```

Arguments

`object`	an `mlmm` object, output of `mlmm`.
`parm`	Not used. For compatibility with the generic method.
`level`	[numeric,0-1] the confidence level of the confidence intervals.
`method`	[character] Should pointwise confidence intervals be output (`"none"`) or simultaneous confidence intervals (`"bonferroni"`, ..., `"fdr"`, `"single-step"`, `"single-step2"`) and/or confidence intervals for pooled linear contrast estimates (`"average"`, `"pool.se"`, `"pool.gls"`, `"pool.gls1"`, `"pool.rubin"`, `"p.rejection"`)?
`ordering`	[character] should the output be ordered by type of parameter (`parameter`) or by model (by). Only relevant for `mlmm` objects.n
`...`	other arguments are passed to `confint.Wald_lmm`.

Details

Statistical inference following pooling is performed according to Rubin's rule whose validity requires the congeniality condition of Meng (1994). **Pooling estimates**: available methods are:

- `"average"`: average estimates
- `"pool.fixse"`: weighted average of the estimates, with weights being the inverse of the squared standard error. The uncertainty about the weights is neglected.
- `"pool.se"`: weighted average of the estimates, with weights being the inverse of the squared standard error. The uncertainty about the weights is computed under independence of the variance parameters between models.
- `"pool.gls"`: weighted average of the estimates, with weights being based on the variance-covariance matrix of the estimates. When this matrix is singular, its spectral decomposition is truncated when the correspodning eigenvalues are below 10^{-12}. The uncertainty about the weights is neglected.
- `"pool.gls1"`: similar to `"pool.gls"` but ensure that the weights are at most 1 in absolute value by shrinking toward the average.
- `"pool.rubin"`: average of the estimates and compute the uncertainty according to Rubin's rule (Barnard et al. 1999).

References

Meng X. L.(1994). Multiple-imputation inferences with uncongenial sources of input. Statist. Sci.9, 538–58.

confint.rbindWald_lmm *Confidence Intervals From Combined Wald Tests Applied to Linear Mixed Models*

Description

Combine pointwise or simultaneous confidence intervals relative to linear contrasts of parameters from different linear mixed models. Can also output p-values (corresponding to pointwise confidence intervals) or adjusted p-values (corresponding to simultaneous confidence intervals).

Usage

```
## S3 method for class 'rbindWald_lmm'
confint(
  object,
  parm,
  level = 0.95,
  df = NULL,
  method = NULL,
  columns = NULL,
  ordering = NULL,
  backtransform = NULL,
  ...
)
```

Arguments

`object`	a `rbindWald_lmm` object.
`parm`	Not used. For compatibility with the generic method.
`level`	[numeric, 0-1] nominal coverage of the confidence intervals.
`df`	[logical] Should a Student's t-distribution be used to model the distribution of the Wald statistic. Otherwise a normal distribution is used.
`method`	[character] Should pointwise confidence intervals be output (`"none"`) or simultaneous confidence intervals (`"bonferroni"`, ..., `"fdr"`, `"single-step"`, `"single-step2"`) and/or confidence intervals for pooled linear contrast estimates (`"average"`, `"pool.se"`, `"pool.gls"`, `"pool.gls1"`, `"pool.rubin"`, `"p.rejection"`)? Only relevant when `effects` = `"Wald"`.
`columns`	[character vector] Columns to be output. Can be any of `"estimate"`, `"se"`, `"statistic"`, `"df"`, `"null"`, `"lower"`, `"upper"`, `"p.value"`.
`ordering`	[character] should the output be ordered by name of the linear contrast (`"contrast"`) or by model (`"model"`).
`backtransform`	[logical] should the estimates be back-transformed?
`...`	Not used. For compatibility with the generic method.

Details

Argument **method**: the following pooling are available:

- `"average"`: average estimates
- `"pool.se"`: weighted average of the estimates, with weights being the inverse of the squared standard error.
- `"pool.gls"`: weighted average of the estimates, with weights being based on the variance-covariance matrix of the estimates. When this matrix is singular, the Moore–Penrose inverse is used which correspond to truncate the spectral decomposition for eigenvalues below 10^{-12}.
- `"pool.gls1"`: similar to `"pool.gls"` with weights shrinked toward the average whenever they exceed 1 in absolute value.
- `"pool.rubin"`: average of the estimates and compute the uncertainty according to Rubin's rule (Barnard et al. 1999). Validity requires the congeniality condition of Meng (1994).
- `"p.rejection"`: proportion of null hypotheses where there is evidence for an effect. By default the critical quantile (defining the level of evidence required) is evaluated using a `"single-step"` method but this can be changed by adding adjustment method in the argument `method`, e.g. `effects=c("bonferronin","p.rejection")`.

`confint.resample` *Resampling Confidence Intervals for Linear Mixed Model*

Description

Compute confidence intervals for linear mixed model using resampling (permutation or bootstrap).

Usage

```
## S3 method for class 'resample'
confint(
  object,
  parm = NULL,
  null = NULL,
  level = 0.95,
  method = NULL,
  columns = NULL,
  correction = TRUE,
  ...
)
```

Arguments

`object`	a `reample` object.
`parm`	Not used. For compatibility with the generic method.
`null`	[numeric vector] the value of the null hypothesis relative to each coefficient. Only relevant for when using bootstrap.

`level` [numeric,0-1] the confidence level of the confidence intervals.

`method` [character] method used to compute the confidence intervals and p-values: `"percentile"`, `"gaussian"`, or `"studentized"`.

`columns` [character vector] Columns to be output. Can be any of `"estimate"`, `"sample.estimate"`, `"se"`, `"sample.se"`, `"null"`, `"lower"`, `"upper"`, `"p.value"`.

`correction` [logical] correction to ensure non-0 p-values when using the percentile method, e.g. with permutations the p.value is evaluated as (#more extreme + 1)/(n.sample + 1) instead of (#more extreme)/(n.sample).

`...` Not used. For compatibility with the generic method.

Details

Argument **correction**: if we denote by `n.sample` the number of permutations that have been performed, having the correction avoids p-values of 0 by ensuring they are at least `as.numeric(correction)/(n.sample+as.n` e.g. at least 0.000999001 for a thousand permutations.

Argument **correction** (bootstrap): percentile confidence intervals are computed based on the quantile of the resampling distribution.
Gaussian confidence intervals and p-values are computed by assuming a normal distribution and estimating its variance based on the bootstrap distribution.
Studentized confidence intervals are computed using quantiles based on the boostrap distribution after it has been centered and rescaled by the point estimate and its standard error Percentile and Studentized p-values are computed by finding the coverage level such that one of the bound of the confidence interval equals the null.

Argument **correction** (bootstrap): Percentile p-values are computed based on the proportion of times the estimate was more extreme than the permutation estimates.
Gaussian p-values are computed by assuming a normal distribution and estimating its variance based on the permutation distribution.
Studentized p-values are computed based on the proportion of times the test statistic was more extreme than the permutation test statistic.
No confidence intervals are provided.

`confint.Wald_lmm` *Confidence Intervals From Wald Tests Applied to a Linear Mixed Model*

Description

Compute pointwise or simultaneous confidence intervals relative to linear contrasts of parameters from a linear mixed model. Pointwise confidence intervals have nominal coverage w.r.t. a single contrast whereas simultaneous confidence intervals have nominal coverage w.r.t. to all contrasts. Can also output p-values (corresponding to pointwise confidence intervals) or adjusted p-values (corresponding to simultaneous confidence intervals).

Usage

```
## S3 method for class 'Wald_lmm'
confint(
  object,
  parm,
  level = 0.95,
  df = NULL,
  method = NULL,
  columns = NULL,
  backtransform = NULL,
  ...
)
```

Arguments

object	a Wald_lmm object
parm	Not used. For compatibility with the generic method.
level	[numeric, 0-1] nominal coverage of the confidence intervals.
df	[logical] Should a Student's t-distribution be used to model the distribution of the Wald statistic. Otherwise a normal distribution is used.
method	[character] Should pointwise confidence intervals be output ("none") or simultaneous confidence intervals ("bonferroni", ..., "fdr", "single-step", "single-step2")?
columns	[character vector] Columns to be output. Can be any of "estimate", "se", "statistic", "df", "null", "lower", "upper", "p.value".
backtransform	[logical] should the estimates, standard errors, and confidence intervals be backtransformed?
...	Not used. For compatibility with the generic method.

Details

Available **methods** are:

- "none", "bonferroni", "single-step2"
- "holm", "hochberg", "hommel", "BH", "BY", "fdr": adjustment performed by [stats::p.adjust()], no confidence interval is computed.
- "single-step", "free", "Westfall", "Shaffer": adjustment performed by [multcomp::glht()], for all but the first method no confidence interval is computed.

Note: method "single-step" adjusts for multiple comparisons using equicoordinate quantiles of the multivariate Student's t-distribution over all tests, instead of the univariate quantiles. It assumes equal degrees-of-freedom in the marginal and is described in section 7.1 of Dmitrienko et al. (2013) under the name single-step Dunnett procedure. The name "single-step" is borrowed from the multcomp package. In the book Bretz et al. (2010) written by the authors of the package, the procedure is refered to as max-t tests which is the terminology adopted in the LMMstar package. When degrees-of-freedom differs between individual hypotheses, method "single-step2" is recommended. It simulates data using copula whose marginal distributions are Student's t-distribution (with possibly different degrees-of-freedom) and elliptical copula with parameters the estimated correlation between the test statistics (via the copula package). It then computes the frequency at which the simulated maximum exceed the observed maximum and appropriate quantile of simulated maximum for the confidence interval.

References

Barnard and Rubin, **Small-sample degrees of freedom with multiple imputation**. *Biometrika* (1999), 86(4):948-955.
Dmitrienko, A. and D'Agostino, R., Sr (2013), **Traditional multiplicity adjustment methods in clinical trials**. *Statist. Med.*, 32: 5172-5218. Frank Bretz, Torsten Hothorn and Peter Westfall (2010), **Multiple Comparisons Using R**, *CRC Press*, Boca Raton.

`correlate` *Compute correlation matrix*

Description

Compute correlation matrix for multiple variables and/or multiple groups.

Usage

```
correlate(
  formula,
  data,
  repetition,
  use = "everything",
  method = "pearson",
  collapse.value = ".",
  collapse.var = ".",
  na.rm
)
```

Arguments

`formula`	[formula] on the left hand side the outcome(s) and on the right hand side the grouping variables. E.g. Y1+Y2 ~ Gender will compute for each gender the correlation matrix for Y1 and the correaltion matrix for Y2.
`data`	[data.frame] dataset containing the observations.
`repetition`	[formula] Specify the structure of the data: the time/repetition variable and the grouping variable, e.g. ~ time|id.
`use`	[character] method for computing correlation in the presence of missing values: `"everything"`, `"all.obs"`, `"complete.obs"`, `"na.or.complete"`, or `"pairwise.complete.obs"`. Passed to `stats::cor`.
`method`	[character] type of correlation coefficient: `"pearson"`, `"kendall"`, `"spearman"`. Passed to `stats::cor`.

`collapse.value` [character] symbol used to combine covariate values when using multiple grouping variables.

`collapse.var` [character] symbol used to combine variable names to be pasted left of the covariate values when using multiple grouping variables. Can be disabled setting it to `NULL` or `FALSE`.

`na.rm` [logical] not used. The user may expect this argument though so it is added to help the user with a message pointing toward the argument use.

See Also

`as.array` to convert the output to an array or `as.matrix` to convert the output to a matrix (when a single outcome and no grouping variable).
`summarize` for other summary statistics (e.g. mean, standard deviation, ...).

Examples

```
#### simulate data (wide format) ####
data(gastricbypassL, package = "LMMstar")

## compute correlations (composite time variable)
e.S <- correlate(weight ~ 1, data = gastricbypassL, repetition = ~time|id)
e.S
as.matrix(e.S)

e.S21 <- correlate(glucagonAUC + weight ~ 1, data = gastricbypassL,
                  repetition = ~time|id, use = "pairwise")
e.S21
as.array(e.S21)
as.matrix(e.S21, index = "weight")

gastricbypassL$sex <- as.numeric(gastricbypassL$id) %% 2
e.S12 <- correlate(weight ~ sex, data = gastricbypassL,
                   repetition = ~time|id, use = "pairwise")
e.S12
as.array(e.S12)

e.S22 <- correlate(glucagonAUC + weight ~ sex, data = gastricbypassL,
                   repetition = ~time|id, use = "pairwise")
e.S22
as.array(e.S22)
```

`CS` *Compound Symmetry Structure*

Description

Variance-covariance structure where the variance and correlation of the residuals is constant within covariate levels. Can be stratified on a categorical variable. The default has no covariate and therefore the variance and correlation are constant within cluster.

Usage

```
CS(formula, var.cluster, var.time, type = NULL, group.type = NULL, add.time)
```

Arguments

`formula` formula indicating on which variable to stratify the residual variance and correlation (left hand side) and variables influencing the residual variance and correlation (right hand side).

`var.cluster` [character] cluster variable.

`var.time` [character] time variable.

`type` [character]

- `"ho"`, `"homo"`, or `"homogeneous"`: constant variance and covariate-specific correlation. Analogous to crossed or nested random effects.
- `"he"`, `"hetero"`, or `"heterogeneous"`: variance and correlation specific to the level of covariates. Can be seen as more flexible crossed or nested random effects model.

`group.type` [integer vector] grouping of the regressor for the correlation structure. A constant value corresponds to nested random effects (default) and a regressor-specific value to crossed random effects

`add.time` not used.

Details

A typical formula would be `~1`, indicating a variance constant over time and the same correlation between all pairs of times.

Value

An object of class `CS` that can be passed to the argument `structure` of the `lmm` function.

Examples

```
## no covariates
CS(~1, var.cluster = "id", var.time = "time")
CS(gender~1, var.cluster = "id", var.time = "time")

## covariates
CS(~time, var.cluster = "id", var.time = "time")
CS(gender~time, var.cluster = "id", var.time = "time")
CS(list(~time,~1), var.cluster = "id", var.time = "time")
CS(list(gender~time,gender~1), var.cluster = "id", var.time = "time")
```

CUSTOM *Custom Structure*

Description

Variance-covariance structure specified by the user.

Usage

```
CUSTOM(
  formula,
  var.cluster,
  var.time,
  FCT.sigma,
  dFCT.sigma = NULL,
  d2FCT.sigma = NULL,
  init.sigma,
  FCT.rho,
  dFCT.rho = NULL,
  d2FCT.rho = NULL,
  init.rho,
  add.time
)
```

Arguments

`formula`	formula indicating variables influencing the residual variance and correlation (right hand side).
`var.cluster`	[character] cluster variable.
`var.time`	[character] time variable.
`FCT.sigma`	[function] take as argument the model parameters, time, and design matrix. Output the vector of residuals standard deviations.
`dFCT.sigma`	[list of vectors] list whose elements are the first derivative of argument `FCT.sigma`.
`d2FCT.sigma`	[list of vectors] list whose elements are the second derivative of argument `FCT.sigma` (no cross-terms).
`init.sigma`	[numeric vector] initial value for the variance parameters.
`FCT.rho`	[function] take as argument the model parameters, time, and design matrix. Output the matrix of residuals correlation.
`dFCT.rho`	[list of matrices] list whose elements are the first derivative of argument `FCT.rho`.
`d2FCT.rho`	[list of matrices] list whose elements are the second derivative of argument `FCT.rho` (no cross-terms).
`init.rho`	[numeric vector] initial value for the correlation parameters.
`add.time`	not used.

Value

An object of class CUSTOM that can be passed to the argument structure of the lmm function.

Examples

```
## Compound symmetry structure
CUSTOM(~1,
       FCT.sigma = function(p,n.time,X){rep(p,n.time)},
       init.sigma = c("sigma"=1),
       dFCT.sigma = function(p,n.time,X){list(sigma = rep(1,n.time))},
       d2FCT.sigma = function(p,n.time,X){list(sigma = rep(0,n.time))},
       FCT.rho = function(p,n.time,X){
           matrix(p,n.time,n.time)+diag(1-p,n.time,n.time)
       },
       init.rho = c("rho"=0.5),
       dFCT.rho = function(p,n.time,X){
            list(rho = matrix(1,n.time,n.time)-diag(1,n.time,n.time))
       },
       d2FCT.rho = function(p,n.time,X){list(rho = matrix(0,n.time,n.time))}
)

## 2 block structure
rho.2block <- function(p,n.time,X){
   rho <- matrix(0, nrow = n.time, ncol = n.time)
   rho[1,2] <- rho[2,1] <- rho[4,5] <- rho[5,4] <- p["rho1"]
   rho[1,3] <- rho[3,1] <- rho[4,6] <- rho[6,4] <- p["rho2"]
   rho[2,3] <- rho[3,2] <- rho[5,6] <- rho[6,5] <- p["rho3"]
   rho[4:6,1:3] <- rho[1:3,4:6] <- p["rho4"]
   return(rho)
}
drho.2block <- function(p,n.time,X){
   drho <- list(rho1 = matrix(0, nrow = n.time, ncol = n.time),
                rho2 = matrix(0, nrow = n.time, ncol = n.time),
                rho3 = matrix(0, nrow = n.time, ncol = n.time),
                rho4 = matrix(0, nrow = n.time, ncol = n.time))
   drho$rho1[1,2] <- drho$rho1[2,1] <- drho$rho1[4,5] <- drho$rho1[5,4] <- 1
   drho$rho2[1,3] <- drho$rho2[3,1] <- drho$rho2[4,6] <- drho$rho2[6,4] <- 1
   drho$rho3[2,3] <- drho$rho3[3,2] <- drho$rho3[5,6] <- drho$rho3[6,5] <- 1
   drho$rho4[4:6,1:3] <- drho$rho4[1:3,4:6] <- 1
   return(drho)
}
d2rho.2block <- function(p,n.time,X){
   d2rho <- list(rho1 = matrix(0, nrow = n.time, ncol = n.time),
                 rho2 = matrix(0, nrow = n.time, ncol = n.time),
                 rho3 = matrix(0, nrow = n.time, ncol = n.time),
                 rho4 = matrix(0, nrow = n.time, ncol = n.time))
   return(d2rho)
}

CUSTOM(~variable,
       FCT.sigma = function(p,n.time,X){rep(p,n.time)},
       dFCT.sigma = function(p,n.time,X){list(sigma=rep(1,n.time))},
       d2FCT.sigma = function(p,n.time,X){list(sigma=rep(0,n.time))},
       init.sigma = c("sigma"=1),
       FCT.rho = rho.2block,
       dFCT.rho = drho.2block,
       d2FCT.rho = d2rho.2block,
       init.rho = c("rho1"=0.25,"rho2"=0.25,"rho3"=0.25,"rho4"=0.25))
```

df.residual.lmm *Residual Degrees-of-Freedom From a Linear Mixed Model.*

Description

Estimate the residual degrees-of-freedom from a linear mixed model.

Usage

```
## S3 method for class 'lmm'
df.residual(object, ...)
```

Arguments

`object`	a `lmm` object.
`...`	Passed to `residuals.lmm`.

Details

The residual degrees-of-freedom is estimated using the sum of squared normalized residuals.

Value

A numeric value

df.residual.mlmm *Residuals Degrees-of-Freedom From From Multiple Linear Mixed Model.*

Description

Combine the residuals degrees-of-freedom from group-specific linear mixed models.

Usage

```
## S3 method for class 'mlmm'
df.residual(object, ...)
```

Arguments

object	a lmm object.
...	Passed to residuals.lmm.

Details

The residual degrees-of-freedom is estimated separately for each model using the sum of squared normalized residuals.

Value

A numeric vector with one element for each model.

dummy.coef.lmm *Extract Mean Coefficients in Original Coding From a Linear Mixed Model*

Description

This expands the mean coefficients of the linear mixed model into one coefficient per level of the original variable, i.e., including the reference level(s) where the fitted coefficients are 0.

Usage

```
## S3 method for class 'lmm'
dummy.coef(object, use.na = FALSE, ...)
```

Arguments

object	a lmm object.
use.na	[logical] Should NA or 0 be used to represent excluded coefficients in singular models.
...	Not used. For compatibility with the generic method.

Examples

```
## simulate data in the long format
set.seed(10)
dL <- sampleRem(100, n.times = 3, format = "long")
dL$group <- as.factor(paste0("d",dL$X1))

## fit mixed model with interaction
eUN.lmm <- lmm(Y ~ visit*group, repetition =~visit|id, data = dL, df = FALSE)
dummy.coef(eUN.lmm)

## fit mixed model with baseline constraint
dL$drug <- dL$group
```

```
dL[dL$visit==1,"drug"] <- "d0"
eUN.clmm <- lmm(Y ~ visit + visit:drug, repetition =~visit|id, data = dL, df = FALSE)
dummy.coef(eUN.clmm)

dL$drug <- factor(dL$group, levels = c("none","d0","d1"))
dL[dL$visit==1,"drug"] <- "none"
eUN.clmm.none <- lmm(Y ~ visit:drug, repetition =~visit|id, data = dL, df = FALSE)
dummy.coef(eUN.clmm.none)
```

effects.lmm *Effects Derived For a Linear Mixed Model*

Description

Estimate average counterfactual outcome or contrast of outcome based on a linear mixed model.

Usage

```
## S3 method for class 'lmm'
effects(
  object,
  variable,
  effects = "identity",
  type = "outcome",
  repetition = NULL,
  conditional = NULL,
  ref.repetition = 1,
  ref.variable = 1,
  newdata = NULL,
  rhs = NULL,
  multivariate = FALSE,
  prefix.time = NULL,
  prefix.var = TRUE,
  sep.var = ",",
  ...
)
```

Arguments

object a `lmm` object.

variable [character/list] exposure variable relative to which the effect should be computed. Can also be a list with two elements: the first being the variable (i.e. a character) and the second the levels or values for this variable to be considered.

effects [character] should the average counterfactual outcome for each variable level be evaluated (`"identity"`)? Or the difference in average counterfactual outcome between each pair of variable level (`"difference"`)?

`type` [character/numeric vector] Possible transformation of the outcome: no transformation (`"outcome"`), change from baseline (`"change"`), area under the outcome curve (`"auc"`), or area under the outcome curve minus baseline (`"auc-b"`). Alternatively can be a numeric vector indicating how to weight each timepoint.

`repetition` [character vector] repetition at which the effect should be assessed. By default it will be assessed at all repetitions.

`conditional` [character/data.frame] variable(s) conditional to which the average conterfactual outcome or treatment effect should be computed. Alternatively can also be a data.frame where each column correspond to a variable and the rows to the level of the variable(s).

`ref.repetition` [numeric or character] index or value of the reference level for the repetition variable. Only relevant when `type` equal to `"change"`. Can be `NA` to evaluate change relative to all possible reference levels.

`ref.variable` [numeric or character] index or value of the reference level for the exposure variable. Only relevant when `effects` equal to `"difference"`. Can be `NA` to evaluate the difference relative to all possible reference levels.

`newdata` [data.frame] a dataset reflecting the covariate distribution relative to which the average outcome or contrast should be computed.

`rhs` [numeric] the right hand side of the hypothesis.

`multivariate` [logical] should a multivariate Wald test be used to simultaneously test all null hypotheses.

`prefix.time` [character] When naming the estimates, text to be pasted before the value of the repetition variable. Only relevant when `type = "aoc"` or `type = "ate"`.

`prefix.var` [logical] When naming the estimates, should the variable name be added or only the value?

`sep.var` [character] When naming the estimates, text to be pasted between the values to condition on. Only relevant when `type = "aoc"` or `type = "ate"`.

`...` Arguments passed to `anova.lmm`.

Details

The uncertainty is quantified assuming the contrast matrix to be a-priori known. Said otherwise the standard error does not account for the uncertainty about the covariate distribution.

Examples

```
#### simulate data in the long format ####
set.seed(10)
dL <- sampleRem(100, n.times = 3, format = "long")

#### Linear Mixed Model ####
eUN.lmm <- lmm(Y ~ visit + X1 + X2 + X5,
               repetition = ~visit|id, structure = "UN", data = dL)

## outcome
e.YbyX1 <- effects(eUN.lmm, variable = "X1")
```

```
e.YbyX1
summary(e.YbyX1)
model.tables(e.YbyX1)
coef(e.YbyX1, type = ”contrast”)
effects(eUN.lmm, effects = ”difference”, variable = ”X1”)
effects(eUN.lmm, effects = ”difference”, variable = ”X1”, repetition = ”3”)

## change
effects(eUN.lmm, type = ”change”, variable = ”X1”)
effects(eUN.lmm, type = ”change”, variable = ”X1”, ref.repetition = 2)
effects(eUN.lmm, type = ”change”, variable = ”X1”, conditional = NULL)
effects(eUN.lmm, type = ”change”, effects = ”difference”, variable = ”X1”)

## auc
effects(eUN.lmm, type = ”auc”, variable = ”X1”)
effects(eUN.lmm, type = ”auc”, effects = ”difference”, variable = ”X1”)

#### fit Linear Mixed Model with interaction ####
dL$X1.factor <- as.factor(dL$X1)
dL$X2.factor <- as.factor(dL$X2)
eUN.lmmI <- lmm(Y ~ visit * X1.factor + X2.factor + X5,
               repetition = ~visit|id, structure = ”UN”, data = dL)

## average counterfactual conditional to a categorical covariate
effects(eUN.lmmI, variable = ”X1.factor”,
        conditional = ”X2.factor”, repetition = ”3”)
effects(eUN.lmmI, type = ”change”, variable = ”X1.factor”,
        conditional = ”X2.factor”, repetition = ”3”)
effects(eUN.lmmI, type = ”auc”, variable = ”X1.factor”, conditional = ”X2.factor”)

## average difference in counterfactual conditional to a categorical covariate
effects(eUN.lmmI, effects = ”difference”, variable = ”X1.factor”,
        conditional = c(”X2.factor”), repetition = ”3”)
effects(eUN.lmmI, effects = ”difference”, type = ”change”, variable = ”X1.factor”,
        conditional = c(”X2.factor”), repetition = ”3”)
effects(eUN.lmmI, effects = ”difference”, type = ”auc”, variable = ”X1.factor”,
        conditional = ”X2.factor”)

## average difference in counterfactual conditional to a covariate
effects(eUN.lmmI, effect = ”difference”, variable = ”X1.factor”,
        conditional = data.frame(X5=0:2), repetition = ”3”)
effects(eUN.lmmI, effect = ”difference”, type = ”change”, variable = ”X1.factor”,
        conditional = data.frame(X5=0:2))
```

`estfun.lmm` *Extract the Score Function for Multcomp*

Description

Extract the Score Function for Multcomp. For internal use.

Usage

```
## S3 method for class 'lmm'
estfun(x, ...)
```

Arguments

`x` a `lmm` object.

`...` Not used. For compatibility with the generic method.

Value

A matrix containing the score function for each model parameter (columns) relative to each cluster (rows).

Examples

```
## simulate data in the long format
set.seed(10)
dL <- sampleRem(100, n.times = 3, format = ”long”)

## fit Linear Mixed Model
eUN.lmm <- lmm(Y ~ X1 + X2 + X5, repetition = ~visit|id, structure = ”UN”, data = dL, df = FALSE)

## test multiple linear hypotheses
if(require(multcomp)){
LMMstar.options(effects = c(”mean”))
e.glht <- multcomp::glht(eUN.lmm)
e.glht$linfct
}
```

`estimate.lmm` *Delta Method for a Linear Mixed Model*

Description

Estimate standard errors, confidence intervals, and p-values for a smooth transformation of parameters from a linear mixed model.

Usage

```
## S3 method for class 'lmm'
estimate(
  x,
  f,
  df = !is.null(x$df) & !robust,
  robust = FALSE,
  type.information = NULL,
  level = 0.95,
  method.numDeriv = NULL,
  average = FALSE,
  transform.sigma = NULL,
  transform.k = NULL,
  transform.rho = NULL,
  ...
)
```

Arguments

`x` a `lmm` object.

`f` [function] function taking as input `p`, the linear mixed model parameters, which outputs the parameter(s) of interest. Can accept extra-arguments.

`df` [logical] Should degree-of-freedom, computed using Satterthwaite approximation, for the parameter of interest be output. Can also be a numeric vector providing providing the degrees-of-freedom relative to each estimate.

`robust` [logical] Should robust standard errors (aka sandwich estimator) be output instead of the model-based standard errors. Can also be `2` compute the degrees-of-freedom w.r.t. robust standard errors instead of w.r.t. model-based standard errors.

`type.information` [character] Should the expected information be used (i.e. minus the expected second derivative) or the observed inforamtion (i.e. minus the second derivative).

`level` [numeric,0-1] the confidence level of the confidence intervals.

`method.numDeriv` [character] method used to approximate the gradient: either `"simple"` or `"Richardson"`. Passed to `numDeriv::jacobian`.

`average` [logical] is the estimand the average output of argument `f`? Otherwise consider each individual output of argument `f`.

`transform.sigma` [character] Transformation used on the variance coefficient for the reference level. One of `"none"`, `"log"`, `"square"`, `"logsquare"` - see details.

`transform.k` [character] Transformation used on the variance coefficients relative to the other levels. One of `"none"`, `"log"`, `"square"`, `"logsquare"`, `"sd"`, `"logsd"`, `"var"`, `"logvar"` - see details.

`transform.rho` [character] Transformation used on the correlation coefficients. One of `"none"`, `"atanh"`, `"cov"` - see details.

`...` extra arguments passed to `f`.

Details

Based a first order delta method to evaluate the variance of the estimate. The derivative of the transformation is evaluated using numerical differentiation (`numDeriv::jacobian`).

Argument **robust**: the Satterhwaite approximation for the degrees-of-freedom of robust standard errors is often unreliable. This is why the default is to use the degrees-of-freedom of the modeled based standard error instead.

Examples

```
if(require(lava) && require(nlme)){

#### Random effect ####
set.seed(10)
dL <- sampleRem(1e2, n.times = 3, format = "long")
e.lmm1 <- lmm(Y ~ X1+X2+X3 + (1|id), repetition = ~visit|id, data = dL)
nlme::ranef(e.lmm1, se = TRUE)
e.ranef <- estimate(e.lmm1, f  = function(object, p){nlme::ranef(object, p = p)})
e.ranef

if(require(ggplot2)){
df.gg <- cbind(index = 1:NROW(e.ranef), e.ranef)
gg.ranef <- ggplot(df.gg, aes(x = index, y=estimate, ymin=lower, ymax = upper))
gg.ranef + geom_point() + geom_errorbar() + ylab("estimated random effect") + xlab("id")
}

#### ANCOVA via mixed model ####
set.seed(10)
d <- sampleRem(1e2, n.time = 2)
e.ANCOVA1 <- lm(Y2~Y1+X1, data = d)

dL2 <- reshape(d, direction = "long", idvar = c("id","X1"),
               timevar = "time", times = c("1","2"), varying = c("Y1","Y2"),
               v.names = "Y")

## estimated treatment effect (no baseline constraint)
e.lmm <- lmm(Y ~ time + time:X1, data = dL2, repetition = ~time|id)

e.delta <- estimate(e.lmm, function(p){
     c(Y1 = p["rho(1,2)"]*p["k.2"],
      X1 = p["time2:X1"]-p["k.2"]*p["rho(1,2)"]*p["time1:X1"])
}) ## same estimate and similar standard errors.
e.delta ## Degrees-of-freedom are a bit off though
cbind(summary(e.ANCOVA1)$coef, df = df.residual(e.ANCOVA1))

## estimated treatment effect (baseline constraint)
dL2$time2 <- as.numeric(dL2$time=="2")
e.lmmC <- lmm(Y ~ time2 + time2:X1, data = dL2, repetition = ~time|id)
e.deltaC <- estimate(e.lmmC, function(p){
       c(Y1 = p["rho(1,2)"]*p["k.2"],
         X1 = p["time2:X1"])
})
e.deltaC ## Degrees-of-freedom are a bit more accurate
}
```

estimate.mlmm *Delta Method for Multiple Linear Mixed Models*

Description

Estimate standard errors, confidence intervals, and p-values for a smooth transformation of parameters from group-specific linear mixed models.

Usage

```
## S3 method for class 'mlmm'
estimate(
  x,
  f,
  df = FALSE,
  robust = FALSE,
  type.information = NULL,
  level = 0.95,
  method.numDeriv = NULL,
  average = FALSE,
  transform.sigma = NULL,
  transform.k = NULL,
  transform.rho = NULL,
  ...
)
```

Arguments

`x` a `mlmm` object.

`f` [function] function taking as input `p`, a vector containing the group-specific linear mixed model parameters, which outputs the parameter(s) of interest. Can accept extra-arguments.

`df` [logical] Should degree-of-freedom, computed using Satterthwaite approximation, for the parameter of interest be output. Can also be a numeric vector providing providing the degrees-of-freedom relative to each estimate.

`robust` [logical] Should robust standard errors (aka sandwich estimator) be output instead of the model-based standard errors. Can also be `2` compute the degrees-of-freedom w.r.t. robust standard errors instead of w.r.t. model-based standard errors.

`type.information` [character] Should the expected information be used (i.e. minus the expected second derivative) or the observed inforamtion (i.e. minus the second derivative).

`level` [numeric,0-1] the confidence level of the confidence intervals.

`method.numDeriv`
: [character] method used to approximate the gradient: either `"simple"` or `"Richardson"`. Passed to `numDeriv::jacobian`.

`average`
: [logical] is the estimand the average output of argument `f`? Otherwise consider each individual output of argument `f`.

`transform.sigma`
: [character] Transformation used on the variance coefficient for the reference level. One of `"none"`, `"log"`, `"square"`, `"logsquare"` - see details.

`transform.k`
: [character] Transformation used on the variance coefficients relative to the other levels. One of `"none"`, `"log"`, `"square"`, `"logsquare"`, `"sd"`, `"logsd"`, `"var"`, `"logvar"` - see details.

`transform.rho`
: [character] Transformation used on the correlation coefficients. One of `"none"`, `"atanh"`, `"cov"` - see details.

`...`
: extra arguments passed to `f`.

`fitted.lmm` *Fitted Outcome Value by a Linear Mixed Model.*

Description

Evaluate the expected outcome conditional on covariates or the expected missing outcome value conditional on observed outcomes and covariates based on a linear mixed model.

Usage

```
## S3 method for class 'lmm'
fitted(
  object,
  newdata = NULL,
  type = "mean",
  se = NULL,
  df = NULL,
  keep.data = NULL,
  format = "long",
  seed = NULL,
  simplify = TRUE,
  ...
)
```

Arguments

`object`
: a `lmm` object.

`newdata`
: [data.frame] the covariate values for each cluster.

`type` [character] by default fitted values are output (`NULL`). Can also output the expected outcome (for missing outcomes) based on covariates and other outcome values from the same cluster (`"impute"`), the change or expected change between baseline and each follow-up (`"change"`), or the area under the curve of the outcome (`"auc"`, require a numeric repetition variable).

`se` [character] passed to `predict.lmm` to evaluate the standard error of the fitted value, expected outcome, change in expected outcome, or area under the curve.

`df` [logical] should a Student's t-distribution be used to model the distribution of the predicted mean. Otherwise a normal distribution is used.

`keep.data` [logical] should the dataset relative to which the predictions are evaluated be output along side the predicted values? Only possible in the long format.

`format` [character] should the prediction be output in a matrix format with clusters in row and timepoints in columns (`"wide"`), or in a data.frame/vector with as many rows as observations (`"long"`)

`seed` [integer, >0] random number generator (RNG) state used when starting imputation. If NULL no state is set.

`simplify` [logical] simplify the data format (vector instead of data.frame) and column names (no mention of the time variable) when possible.

`...` additional argument passed the `predict.lmm`.

Details

Essentially a wrapper function of `predict.lmm` where the fitted value are stored in the outcome column of the dataset instead of in a separate column.

Value

When `format="wide"`, a data.frame with as many rows as clusters. When `format="long"`, a data.frame with as many rows as observations (`keep.data==TRUE`) or a vector of length the number of observations (`keep.data==TRUE`).

Examples

```
#### single arm trial ####
data(gastricbypassL, package = "LMMstar")
gastricbypassL <- gastricbypassL[order(gastricbypassL$id,gastricbypassL$visit),]
gastricbypassL$weight0 <- unlist(tapply(gastricbypassL$weight,gastricbypassL$id,
function(x){rep(x[1],length(x))}))

eUN.lmm <- lmm(glucagonAUC ~ visit + weight0, repetition = ~visit|id,
             data = gastricbypassL, df = FALSE)

## fitted mean (conditional on covariates only)
fitted(eUN.lmm)
fitted(eUN.lmm, newdata = data.frame(visit = "3", weight0 = 0))
fitted(eUN.lmm, newdata = data.frame(visit = "3", weight0 = 0),
       keep.data = TRUE)
```

```
## fitted outcome value (conditional on covariates and covariates)
fitted(eUN.lmm, type = "outcome")
gastricbypassL.O <- fitted(eUN.lmm, type = "outcome", keep.data = TRUE)

if(require(ggplot2)){
gg.outcome <- ggplot(gastricbypassL.O,
                     aes(x=time, y = glucagonAUC, color = impute, group = id))
gg.outcome <- gg.outcome + geom_point() + geom_line()## + facet_wrap(~id)
gg.outcome
}

tapply(gastricbypassL.O$glucagonAUC, gastricbypassL.O$time, mean)
effects(eUN.lmm, variable = NULL)

## fitted change value (conditional on covariates and covariates)
gastricbypassL.C <- fitted(eUN.lmm, type = "change", keep.data = TRUE)

if(require(ggplot2)){
gg.change <- ggplot(gastricbypassL.C,
                    aes(x=time, y = glucagonAUC, color = impute, group = id))
gg.change <- gg.change + geom_point() + geom_line()
gg.change
}

tapply(gastricbypassL.C$glucagonAUC, gastricbypassL.O$time, mean)
effects(eUN.lmm, type = "change", variable = NULL)

## fitted auc (conditional on covariates and covariates)
gastricbypassL.AUC <- fitted(eUN.lmm, type = "auc", keep.data = TRUE)

if(require(ggplot2)){
gg.auc <- ggplot(gastricbypassL.AUC,
                    aes(x = "auc", y = glucagonAUC, color = impute))
gg.auc <- gg.auc + geom_point()
gg.auc
}

mean(gastricbypassL.AUC$glucagonAUC)
effects(eUN.lmm, type = "auc", variable = NULL)

#### two arm trial ####
## Not run:
if(require(nlmeU) & require(reshape2)){
data(armd.wide, package = "nlmeU")
armd.long <- melt(armd.wide,
                  measure.vars = paste0("visual",c(0,4,12,24,52)),
                  id.var = c("subject","lesion","treat.f","miss.pat"),
                  variable.name = "week",
                  value.name = "visual")

armd.long$week <- factor(armd.long$week,
                         level = paste0("visual",c(0,4,12,24,52)),
                         labels = c(0,4,12,24,52))
eUN2.lmm <- lmm(visual ~ treat.f*week + lesion,
               repetition = ~week|subject, structure = "UN",
               data = armd.long)

## fitted outcome value (conditional on covariates and covariates)
armd.O <- fitted(eUN2.lmm, type = "outcome", keep.data = TRUE)

gg2.outcome <- ggplot(armd.O,
                     aes(x=week, y = visual, color = impute, group = subject))
gg2.outcome <- gg2.outcome + geom_point() + geom_line() + facet_wrap(~treat.f)
gg2.outcome

aggregate(visual ~ week + treat.f, FUN = mean, data = armd.O)
effects(eUN2.lmm, variable = "treat.f") ## mismatch due to adjustment on lesion

## fitted change value (conditional on covariates and covariates)
armd.C <- fitted(eUN2.lmm, type = "change", keep.data = TRUE)

gg.change <- ggplot(armd.C,
                    aes(x=week, y = visual, color = impute, group = subject))
gg.change <- gg.change + geom_point() + geom_line() + facet_wrap(~treat.f)
gg.change

coef(eUN2.lmm)
effects(eUN2.lmm, type = "change", variable = "treat.f")
effects(eUN2.lmm, type = c("change","difference"), variable = "treat.f")

## fitted auc (conditional on covariates and covariates)
armd.AUC <- fitted(eUN2.lmm, type = "auc", keep.data = TRUE)

gg.auc <- ggplot(armd.AUC, aes(x = treat.f, y = visual, color = impute))
gg.auc <- gg.auc + geom_point()
gg.auc

aggregate(visual ~ treat.f, data = armd.AUC, FUN = "mean")
effects(eUN2.lmm, type = "auc", variable = "treat.f") ## adjusted for lesion
effects(eUN2.lmm, type = c("auc","difference"), variable = "treat.f")
}
## End(Not run)
```

fitted.mlmm *Fitted Outcome Value by Multiple Linear Mixed Models.*

Description

Evaluate the expected mean conditional to covariates or the expected missing outcome values conditional to observed outcome values and covariates. based on group-specific linear mixed models.

Usage

```
## S3 method for class 'mlmm'
fitted(object, newdata = NULL, simplify = TRUE, ...)
```

Arguments

`object` a `mlmm` object.

`newdata` [data.frame] the covariate values for each cluster along with the by variable indicating which linear mixed model should be used.

`simplify` [logical] simplify the column names (no mention of the time variable) and remove the cluster, by, and time column when possible.

`...` additional argument passed the `predict.lmm`.

`gastricbypassL` *Data From The Gastric Bypass Study (Long Format)*

Description

Data from the gastric bypass study where the bodyweight and serum glucagon (a gut hormone) were measured in 20 obese subjects prior and after gastric bypass surgery. This dataset is in the long format (i.e. one line per measurement).

- `id`: patient identifier.
- `visit`: the visit index (factor).
- `time`: the week at which the visit took place (numeric).
- `weight`: bodyweight (in kg) measured during the visit.
- `glucagonAUC`: glucagon measured during the visit (in pmol/l x hours).

Usage

```
data(gastricbypassL)
```

References

The effect of Roux-en-Y gastric bypass surgery on the gut mucosal gene expression profile and circulating gut hormones. `https://easddistribute.m-anage.com/from.storage?image=4iBH9mRQm1kfeEHULC2Cxovdly`

gastricbypassW *Data From The Gastric Bypass Study (Wide Format)*

Description

Data from the gastric bypass study where the bodyweight and serum glucagon (a gut hormone) were measured in 20 obese subjects prior and after gastric bypass surgery. This dataset is in the wide format (i.e. one line per patient).

- `id`: patient identifier.
- `weight1`: bodyweight (in kg) 3 months before surgery.
- `weight2`: bodyweight (in kg) 1 week before surgery.
- `weight3`: bodyweight (in kg) 1 week after surgery.
- `weight4`: bodyweight (in kg) 3 months after surgery.
- `glucagonAUC1`: glucagon (in pmol/l x hours) 3 months before surgery.
- `glucagonAUC2`: glucagon (in pmol/l x hours) 1 week before surgery.
- `glucagonAUC3`: glucagon (in pmol/l x hours) 1 week after surgery.
- `glucagonAUC4`: glucagon (in pmol/l x hours) 3 months after surgery.

Usage

```
data(gastricbypassW)
```

References

The effect of Roux-en-Y gastric bypass surgery on the gut mucosal gene expression profile and circulating gut hormones. https://easddistribute.m-anage.com/from.storage?image=4iBH9mRQm1kfeEHULC2Cxovdly

ID *identity Structure*

Description

Variance-covariance structure where the residuals are independent and identically distributed. Can be stratified on a categorical variable.

Usage

```
ID(formula, var.cluster, var.time, add.time)
```

Arguments

formula	formula indicating on which variable to stratify the residual variance (left hand side).
var.cluster	[character] cluster variable.
var.time	[character] time variable.
add.time	not used.

Details

A typical formula would be `~1`.

Value

An object of class `IND` that can be passed to the argument `structure` of the `lmm` function.

Examples

```
ID(NULL, var.cluster = "id", var.time = "time")
ID(~1, var.cluster = "id", var.time = "time")
ID(~gender, var.cluster = "id", var.time = "time")
ID(gender~1, var.cluster = "id", var.time = "time")
```

iid.lmm *Extract the Influence Function From a Linear Mixed Model*

Description

Extract the influence function for an ML or REML estimator of parameters from a linear mixed model.

Usage

```
## S3 method for class 'lmm'
iid(
  x,
  effects = "mean",
  p = NULL,
  robust = TRUE,
  type.information = NULL,
  transform.sigma = NULL,
  transform.k = NULL,
  transform.rho = NULL,
  transform.names = TRUE,
  REML2ML = NULL,
  ...
)
```

Arguments

`x` a `lmm` object.

`effects` [character] Should the influence function for all coefficients be output (`"all"`), or only for coefficients relative to the mean (`"mean"` or `"fixed"`), or only for coefficients relative to the variance structure (`"variance"`), or only for coefficients relative to the correlation structure (`"correlation"`). Can also contain `"gradient"` to also output the gradient of the influence function.

`p` [numeric vector] value of the model coefficients at which to evaluate the influence function. Only relevant if differs from the fitted values.

`robust` [logical] If `FALSE` the influence function is rescaled to match the model-based standard errors. The correlation however will not (necessarily) match the model-based correlation.

`type.information` [character] Should the expected information be used (i.e. minus the expected second derivative) or the observed inforamtion (i.e. minus the second derivative).

`transform.sigma` [character] Transformation used on the variance coefficient for the reference level. One of `"none"`, `"log"`, `"square"`, `"logsquare"` - see details.

`transform.k` [character] Transformation used on the variance coefficients relative to the other levels. One of `"none"`, `"log"`, `"square"`, `"logsquare"`, `"sd"`, `"logsd"`, `"var"`, `"logvar"` - see details.

`transform.rho` [character] Transformation used on the correlation coefficients. One of `"none"`, `"atanh"`, `"cov"` - see details.

`transform.names` [logical] Should the name of the coefficients be updated to reflect the transformation that has been used?

`...` Not used. For compatibility with the generic method.

Details

The influence function equals the individual score rescaled by the (inverse) information. With the expected information and for a lmm fitted using ML, the information is block diagonal so the influence function for the mean and variance parameters can be computed separately. Otherwise the information and individual score relative to all model parameters should be considered. The later is probablematic when using REML as the REML term is the ratio of two term linear in the individual contributions which is not itself linear in the individual contributions. As an add-hoc solution, the denominator is treated as fixed so the ratio is decomposed w.r.t. its numerator.

Value

A matrix with one row per observation and one column per parameter.

- `df=TRUE`: with an attribute `"df"` containing a numeric vector with one element per parameter.
- `effects` includes `"gradient"`: with an attribute `"gradient"` containing a 3 dimensional array with dimension the number of parameters.

Examples

```
data(gastricbypassL)

#### Case WITHOUT cross terms ####
e.lmmREML <- lmm(weight ~ visit, method.fit = "REML", df = FALSE,
                 repetition = ~ visit|id, data = gastricbypassL)
e.lmmML <- lmm(weight ~ visit, method.fit = "ML", df = FALSE,
              repetition = ~ visit|id, data = gastricbypassL)

name.mu <- names(coef(e.lmmREML, effects = "mean"))
name.sigmakrho <- names(coef(e.lmmREML, effects = c("variance","correlation")))
info.REML <- information(e.lmmREML, effects = "all", transform.names = FALSE)
info.ML <- information(e.lmmML, effects = "all", transform.names = FALSE)
info.REML2ML <- information(e.lmmML, p = coef(e.lmmREML, effects = "all"),
                            effects = "all", transform.names = FALSE)

range(info.REML[name.mu,name.sigmakrho])
range(info.ML[name.mu,name.sigmakrho])
range(info.REML[name.mu,]-info.REML2ML[name.mu,])
range(iid(e.lmmREML, REML2ML = TRUE) - iid(e.lmmREML, REML2ML = FALSE))
## neglectable differences

#### Case WITH cross terms ####
e2.lmmREML <- lmm(glucagonAUC ~ visit + weight, method.fit = "REML", df = FALSE,
                 repetition = ~ visit|id, data = gastricbypassL)
e2.lmmML <- lmm(glucagonAUC ~ visit + weight, method.fit = "ML", df = FALSE,
              repetition = ~ visit|id, data = gastricbypassL)

name2.mu <- names(coef(e2.lmmREML, effects = "mean"))
name2.sigmakrho <- names(coef(e2.lmmREML, effects = c("variance","correlation")))
info2.REML <- information(e2.lmmREML, effects = "all", transform.names = FALSE)
info2.ML <- information(e2.lmmML, effects = "all", transform.names = FALSE)
info2.REML2ML <- information(e2.lmmML, p = coef(e2.lmmREML, effects = "all"),
                            effects = "all", transform.names = FALSE)

range(info2.REML[name.mu,]-info2.REML2ML[name.mu,])
## neglectable difference
range(info2.REML[name.mu,name.sigmakrho])
range(info2.ML[name.mu,name.sigmakrho])
range(iid(e2.lmmREML, REML2ML = TRUE) - iid(e2.lmmREML, REML2ML = FALSE))
## non-neglectable differences
diag(crossprod(iid(e2.lmmREML, REML2ML = TRUE)))/diag(vcov(e2.lmmREML))
diag(crossprod(iid(e2.lmmREML, REML2ML = FALSE)))/diag(vcov(e2.lmmREML))
```

`iid.mlmm` *Extract the Influence Function From Multiple Linear Mixed Models*

Description

Extract the influence function for ML or REML estimators of parameters from group-specific linear mixed models.

Usage

```
## S3 method for class 'mlmm'
iid(x, effects = "contrast", p = NULL, ordering = "by", simplify = TRUE, ...)
```

---

iid.Wald_lmm *Extract the Influence Function From Wald Tests Applied to a Linear Mixed Model*

Description

Extract the influence function of linear contrasts applied to an ML or REML estimator of parameters from a linear mixed model.

Usage

```
## S3 method for class 'Wald_lmm'
iid(x, effects = "Wald", transform.names = TRUE, ...)
```

Arguments

effects	[character] should the influence function for the linear contrasts involved in the Wald test be output ("Wald"), or for the linear mixed model parameters ("all")?
transform.names	[logical] should the name of the coefficients be updated to reflect the transformation that has been used? Only relevant when effects="all".
...	Not used. For compatibility with the generic method.
object	a Wald_lmm object.

Value

A matrix with one row per observation and one column per parameter (effects="Wald" or effects="all") or a logical value (effects="test").

IND *Independence Structure*

Description

Variance-covariance structure where the residuals are independent but may have different variance. Can be stratified on a categorical variable.

Usage

```
IND(formula, var.cluster, var.time, add.time)
```

Arguments

formula	formula indicating variables influencing the residual variance, using either as a multiplicative factor (right hand side) or stratification (left hand side) to model their effect.
var.cluster	[character] cluster variable.
var.time	[character] time variable.
add.time	Should the default formula (i.e. when NULL) contain a time effect.

Details

A typical formula would be either ~1 indicating constant variance or ~time indicating a time dependent variance.

Value

An object of class IND that can be passed to the argument structure of the lmm function.

Examples

```
IND(NULL, var.cluster = "id", var.time = "time", add.time = TRUE)
IND(~1, var.cluster = "id", var.time = "time")
IND(gender~1, var.cluster = "id", var.time = "time")
IND(gender~time, var.cluster = "id", var.time = "time")
IND(~gender+time, var.cluster = "id", var.time = "time")
```

information.lmm *Extract The Information From a Linear Mixed Model*

Description

Extract or compute minus the second derivative of the log-likelihood of a linear mixed model.

Usage

```
## S3 method for class 'lmm'
information(
  x,
  effects = NULL,
  newdata = NULL,
  p = NULL,
  indiv = FALSE,
  type.information = NULL,
  transform.sigma = NULL,
  transform.k = NULL,
  transform.rho = NULL,
  transform.names = TRUE,
  ...
)
```

Arguments

x a lmm object.

effects [character] Should the information relative to all coefficients be output ("all" or "fixed"), or only coefficients relative to the mean ("mean"), or only coefficients relative to the variance and correlation structure ("variance" or "correlation").

newdata [data.frame] dataset relative to which the information should be computed. Only relevant if differs from the dataset used to fit the model.

p [numeric vector] value of the model coefficients at which to evaluate the information. Only relevant if differs from the fitted values.

indiv [logical] Should the contribution of each cluster to the information be output? Otherwise output the sum of all clusters of the derivatives.

type.information [character] Should the expected information be computed (i.e. minus the expected second derivative) or the observed inforamtion (i.e. minus the second derivative).

transform.sigma [character] Transformation used on the variance coefficient for the reference level. One of "none", "log", "square", "logsquare" - see details.

transform.k [character] Transformation used on the variance coefficients relative to the other levels. One of "none", "log", "square", "logsquare", "sd", "logsd", "var", "logvar" - see details.

transform.rho [character] Transformation used on the correlation coefficients. One of "none", "atanh", "cov" - see details.

transform.names [logical] Should the name of the coefficients be updated to reflect the transformation that has been used?

... Not used. For compatibility with the generic method.

Details

For details about the arguments **transform.sigma**, **transform.k**, **transform.rho**, see the documentation of the coef.lmm function.

Value

When argument indiv is FALSE, a matrix with the value of the infroamtion relative to each pair of coefficient (in rows and columns) and each cluster (in rows). When argument indiv is TRUE, a 3-dimensional array with the value of the information relative to each pair of coefficient (dimension 2 and 3) and each cluster (dimension 1).

levels.lmm *Contrasts and Reference Level in a Linear Mixed Model*

Description

Contrasts and reference level used when modeling the mean in a linear mixed modek.

Usage

```
## S3 method for class 'lmm'
levels(x)
```

Arguments

`x`	an `lmm` object.

Value

a list with two elements

- all: contrast matrix for each categorical or factor variable
- reference: reference level: one value for each categorical variable

lmm *Fit Linear Mixed Model*

Description

Fit a linear mixed model defined by a mean and a covariance structure.

Usage

```
lmm(
  formula,
  data,
  repetition,
  structure,
  weights = NULL,
  method.fit = NULL,
  df = NULL,
  type.information = NULL,
  trace = NULL,
  control = NULL
)
```

Arguments

`formula` [formula] Specify the model for the mean. On the left hand side the outcome and on the right hand side the covariates affecting the mean value. E.g. Y ~ Gender + Gene.

`data` [data.frame] dataset (in the long format) containing the observations.

`repetition` [formula] Specify the structure of the data: the time/repetition variable and the grouping variable, e.g. ~ time|id.

`structure` [character] type of covariance structure, either `"CS"` (compound symmetry) or `"UN"` (unstructured).

`weights` [formula or character] variable in the dataset used to weight the log-likelihood and its derivative. Should be constant within cluster.

`method.fit` [character] Should Restricted Maximum Likelihoood (`"REML"`) or Maximum Likelihoood (`"ML"`) be used to estimate the model parameters?

`df` [logical] Should the degrees-of-freedom be computed using a Satterthwaite approximation?

`type.information` [character] Should the expected information be computed (i.e. minus the expected second derivative) or the observed inforamtion (i.e. minus the second derivative).

`trace` [interger, >0] Show the progress of the execution of the function.

`control` [list] Control values for the optimization method. The element `optimizer` indicates which optimizer to use and additional argument will be pass to the optimizer.

Details

Computation time the `lmm` has not been developped to be a fast function as, by default, it uses REML estimation with the observed information matrix and uses a Satterthwaite approximation to compute degrees-of-freedom (this require to compute the third derivative of the log-likelihood which is done by numerical differentiation). The computation time can be substantially reduced by using ML estimation with the expected information matrix and no calculation of degrees-of-freedom: arguments `method.fit="ML"`, `type.information="expected"`, `df=FALSE`. This will, however, lead to less accurate p-values and confidence intervals in small samples.

By default, the estimation of the model parameters will be made using a Newton Raphson algorithm. This algorithm does not ensure that the residual covariance matrix is positive definite and therefore may sometimes fail. See argument optimizer in `LMMstar.options`.

Argument control: when using the optimizer `"FS"`, the following elements can be used

- `init`: starting values for the model parameters.
- `n.iter`: maximum number of interations of the optimization algorithm.
- `tol.score`: score value below which convergence has been reached.
- `tol.param`: difference in estimated parameters from two successive iterations below which convergence has been reached.
- `trace`: display progress of the optimization procedure.

Argument repetition: when numeric, it will be converted into a factor variable, possibly adding a leading 0 to preserve the ordering. This transformation may cause inconsistency when combining results between different lmm object. This is why the grouping variable should preferably be of type character or factor.

Value

an object of class lmm containing the estimated parameter values, the residuals, and relevant derivatives of the likelihood.

See Also

summary.lmm for a summary of the model fit.
model.tables.lmm for a data.frame containing estimates with their uncertainty.
plot.lmm for a graphical display of the model fit or diagnostic plots.
levels.lmm to display the reference level.
anova.lmm for testing linear combinations of coefficients (F-test, multiple Wald tests)
effects.lmm for evaluating average marginal or counterfactual effects
sigma.lmm for extracting estimated residual variance-covariance matrices.
residuals.lmm for extracting residuals or creating residual plots (e.g. qqplots).
predict.lmm for evaluating mean and variance of the outcome conditional on covariates or other outcome values.
ranef.lmm for evaluating random effects and variance components in random effects models.

Examples

```
#### 1- simulate data in the long format ####
set.seed(10)
dL <- sampleRem(100, n.times = 3, format = "long")
dL$X1 <- as.factor(dL$X1)
dL$X2 <- as.factor(dL$X2)

#### 2- fit Linear Mixed Model ####
eCS.lmm <- lmm(Y ~ X1 * X2 + X5, repetition = ~visit|id, structure = "CS", data = dL)

logLik(eCS.lmm) ## -670.9439
summary(eCS.lmm)


#### 3- estimates ####
## reference level
levels(eCS.lmm)$reference
## mean parameters
coef(eCS.lmm)
model.tables(eCS.lmm)
confint(eCS.lmm)

## all parameters
coef(eCS.lmm, effects = "all")
model.tables(eCS.lmm, effects = "all")
confint(eCS.lmm, effects = "all")
```

```
## variance-covariance structure
sigma(eCS.lmm)

#### 4- diagnostic plots ####
quantile(residuals(eCS.lmm))
quantile(residuals(eCS.lmm, type = ”normalized”))

## Not run:
if(require(ggplot2)){
  ## investigate misspecification of the mean structure
  plot(eCS.lmm, type = ”scatterplot”)
  ## investigate misspecification of the variance structure
  plot(eCS.lmm, type = ”scatterplot2”)
  ## investigate misspecification of the correlation structure
  plot(eCS.lmm, type = ”correlation”)
  ## investigate misspecification of the residual distribution
  plot(eCS.lmm, type = ”qqplot”)
}

## End(Not run)

#### 5- statistical inference ####
anova(eCS.lmm) ## effect of each variable
anova(eCS.lmm, effects = ”X11-X21=0”) ## test specific coefficient
## test several hypothese with adjustment for multiple comparisons
summary(anova(eCS.lmm, effects = c(”X11=0”,”X21=0”)))

#### 6- prediction ####
## conditional on covariates
newd <- dL[1:3,]
predict(eCS.lmm, newdata = newd, keep.data = TRUE)
## conditional on covariates and outcome
newd <- dL[1:3,]
newd$Y[3] <- NA
predict(eCS.lmm, newdata = newd, type = ”dynamic”, keep.data = TRUE)

#### EXTRA ####
if(require(mvtnorm)){
## model for the average over m replicates
## (only works with independent replicates)
Sigma1 <- diag(1,1,1); Sigma5 <- diag(1,5,5)
n <- 100
dfW <- rbind(data.frame(id = 1:n, rep = 5, Y = rowMeans(rmvnorm(n, sigma = Sigma5))),
             data.frame(id = (n+1):(2*n), rep = 1, Y = rmvnorm(n, sigma = Sigma1)))

e.lmmW <- lmm(Y~1, data = dfW, weigths=~rep, control = list(optimizer = ”FS”))
e.lmm0 <- lmm(Y~1, data = dfW, control = list(optimizer = ”FS”))
model.tables(e.lmmW, effects = ”all”)
model.tables(e.lmm0, effects = ”all”)
## TRUE standard error is 1

}
```

lmmCC *Fit Linear Mixed Model on Complete Case data*

Description

Fit a linear mixed model on the complete case data. Mostly useful as a sanity check, to match the results of a univariate analysis on the change.

Usage

```
lmmCC(object, ...)

## S3 method for class 'formula'
lmmCC(
  object,
  repetition,
  data,
  lm.change = FALSE,
  df = NULL,
  trace = TRUE,
  control = NULL,
  ...
)

## S3 method for class 'lm'
lmmCC(
  object,
  repetition,
  data,
  name.time = "time",
  df = NULL,
  trace = TRUE,
  control = NULL,
  ...
)
```

Arguments

`object` [formula] Specify the model for the mean. On the left hand side the outcome and on the right hand side the covariates affecting the mean value. E.g. Y ~ Gender + Gene.

`...` Not used. For compatibility with the generic method.

`repetition` [formula] Specify the structure of the data: the time/repetition variable and the grouping variable, e.g. ~ time|id.

`data` [data.frame] dataset (in the long format) containing the observations.

`lm.change` [logical] Should a linear model on the change in outcome be estimated. Only possible with two repetitions. Will match the mixed model if the later includes repetition-dependent effects for all covariates.

`df` [logical] Should the degrees-of-freedom be computed using a Satterthwaite approximation?

`trace` [interger, >0] Show the progress of the execution of the function.

`control` [list] Control values for the optimization method. The element `optimizer` indicates which optimizer to use and additional argument will be pass to the optimizer.

`name.time` [character] name of the time variable.

Value

A `lmmCC` object, which inherits from `lmm`.

Examples

```
#### 1- simulate data in the wide format ####
set.seed(10)
dW <- sampleRem(100, n.times = 3, format = "wide")
dW$Y3[1:10] <- NA
dW$change2 <- dW$Y2 - dW$Y1
dW$change3 <- dW$Y3 - dW$Y1

e.lm2 <- lm(change2 ~ X1 + X2, data = dW)
summary(e.lm2)$coef
e.lm3 <- lm(change3 ~ X1 + X2, data = dW)
summary(e.lm3)$coef

#### 2- complete case LMM from LM ####
e.lmmCC2 <- lmmCC(e.lm2, repetition = change2~Y2-Y1)
model.tables(e.lmmCC2)
e.lmmCC3 <- lmmCC(e.lm3, repetition = change3~Y3-Y1)
model.tables(e.lmmCC3)

#### 3- complete case LMM ####
dL <- reshape(dW[,c("id","X1","X2","Y1","Y2","Y3")],
              direction = "long",
              varying = c("Y1","Y2","Y3"), sep = "", idvar = "id")
dL$time <- as.character(dL$time)

e.lmm2 <- lmmCC(Y ~ time + time*X1 + time*X2, repetition = ~time|id,
                data = dL[dL$time %in% 1:2,])
model.tables(e.lmm2)
e.lmm3.bis <- lmmCC(Y ~ time + time*X1 + time*X2, repetition = ~time|id,
                data = dL[dL$time %in% c(1,3),], lm.change = TRUE)
model.tables(e.lmm3.bis)
e.lmm3.bis$lm.change
```

LMMstar.options *Global options for LMMstar package*

Description

Update or select global options for the LMMstar package.

Usage

```
LMMstar.options(..., reinitialise = FALSE)
```

Arguments

`...`	options to be selected or updated
`reinitialise`	should all the global parameters be set to their default value

Details

The options are:

- adj.method [character vector]: possible methods to adjust for multiple comparisons. NOT MEANT TO BE CHANGED BY THE USER.
- backtransform.confint [logical]: should variance/covariance/correlation estimates be back-transformed when they are transformed on the log or atanh scale. Used by `confint`.
- columns.anova [character vector]: columns to ouput when using `anova` with argument `ci=TRUE`.
- columns.confint [character vector]: columns to ouput when using `confint`.
- columns.summarize [character vector]: columns to ouput when displaying descriptive statistics using `summarize`.
- columns.summary [character vector]: columns to ouput when displaying the model coefficients using `summary`.
- df [logical]: should approximate degrees-of-freedom be computed for Wald and F-tests. Used by `lmm`, `anova`, `predict`, and `confint`.
- drop.X [logical]: should columns causing non-identifiability of the model coefficients be dropped from the design matrix. Used by `lmm`.
- effects [character]: parameters relative to which estimates, score, information should be output.
- min.df [integer]: minimum possible degree-of-freedom. Used by `confint`.
- method.fit [character]: objective function when fitting the Linear Mixed Model (REML or ML). Used by `lmm`.
- method.numDeriv [character]: type used to approximate the third derivative of the log-likelihood (when computing the degrees-of-freedom). Can be `"simple"` or `"Richardson"`. See `numDeriv::jacobian` for more details. Used by `lmm`.
- n.sampleCopula [integer]: number of samples used to compute confidence intervals and p-values adjusted for multiple comparisons via `"single-step2"`. Used by `confint.Wald_lmm`.

- optimizer [character]: method used to estimate the model parameters. Either `"FS"`, an home-made fisher scoring algorithm, or a method from `optimx:optimx` like `"BFGS"`or `Nelder-Mead`.
- param.optimizer [numeric vector]: default option for the `FS` optimization routine: maximum number of gradient descent iterations (`n.iter`), maximum acceptable score value (`tol.score`), maximum acceptable change in parameter value (`tol.param`).
- pool.method [character vector]: possible methods to pool estimates. NOT MEANT TO BE CHANGED BY THE USER.
- precompute.moments [logical]: Should the cross terms between the residuals and design matrix be pre-computed. Useful when the number of subject is substantially larger than the number of mean paramters.
- sep [character vector]: character used to combined two strings of characters in various functions (lp: .vcov.model.matrix, k.cov/k.strata: .skeletonK, pattern: .findUpatterns, rho.name/rho.strata: .skeletonRho, reformat: .reformat).
- trace [logical]: Should the progress of the execution of the `lmm` function be displayed?
- tranform.sigma, tranform.k, tranform.rho: transformation used to compute the confidence intervals/p-values for the variance and correlation parameters. See the detail section of the coef function for more information. Used by `lmm`, `anova` and `confint`.
- type.information [character]: Should the expected or observed information (`"expected"` or `"observed"`) be used to perform statistical inference? Used by `lmm`, `anova` and `confint`.

Value

A list containing the default options.

`logLik.lmm` *Log-Likelihood From a Linear Mixed Model*

Description

Extract or compute the log-likelihood of a linear mixed model.

Usage

```
## S3 method for class 'lmm'
logLik(object, newdata = NULL, p = NULL, indiv = FALSE, ...)
```

Arguments

`object`	a `lmm` object.
`newdata`	[data.frame] dataset relative to which the log-likelihood should be computed. Only relevant if differs from the dataset used to fit the model.
`p`	[numeric vector] value of the model coefficients at which to evaluate the log-likelihood. Only relevant if differs from the fitted values.
`indiv`	[logical] Should the contribution of each cluster to the log-likelihood be output? Otherwise output the sum of all clusters of the derivatives.
`...`	Not used. For compatibility with the generic method.

Details

indiv: only relevant when using maximum likelihood. Must be `FALSE` when using restricted maximum likelihood.

Value

A numeric value (total logLikelihood) or a vector of numeric values, one for each cluster (cluster specific logLikelihood).

`logLik.mlmm` *Log-Likelihood From Multiple Linear Mixed Models*

Description

Extract or compute the log-likelihood from each group-specific Linear Mixed Model.

Usage

```
## S3 method for class 'mlmm'
logLik(object, newdata = NULL, p = NULL, indiv = FALSE, ...)
```

Arguments

`object`	a `mlmm` object.
`newdata`	[data.frame] dataset relative to which the log-likelihood should be computed. Only relevant if differs from the dataset used to fit the model.
`p`	[list of numeric vector] values for the model parameters at which to evaluate the log-likelihood. Only relevant if differs from the fitted values.
`indiv`	[logical] Should the contribution of each cluster to the log-likelihood be output? Otherwise output the sum of all clusters of the derivatives.
`...`	Not used. For compatibility with the generic method.

Details

indiv: only relevant when using maximum likelihood. Must be `FALSE` when using restricted maximum likelihood.

Value

A numeric vector when `indiv=FALSE` (log-likelihood per model) or a matrix of numeric values when when `indiv=TRUE` with one line for each cluster and one column for each model.

mlmm *Fit Multiple Linear Mixed Model*

Description

Fit several linear mixed models, extract relevant coefficients, and combine them into a single table.

Usage

```
mlmm(
  ...,
  data,
  by,
  contrast.rbind = NULL,
  effects = NULL,
  robust = FALSE,
  df = NULL,
  name.short = TRUE,
  transform.sigma = NULL,
  transform.k = NULL,
  transform.rho = NULL,
  transform.names = TRUE,
  trace = TRUE
)
```

Arguments

`...`	arguments passed to `lmm`.
`data`	[data.frame] dataset (in the long format) containing the observations.
`by`	[character] variable used to split the dataset. On each split a seperate linear mixed model is fit.
`contrast.rbind`	[character or numeric matrix] Contrast to be be applied to compare the groups. Argument passed to the argument `effects` of `rbind.Wald_lmm`. Right hand side can be specified via an attribute `"rhs"`.
`effects`	[character or numeric matrix] Linear combinations of coefficients relative to which Wald test should be computed. Argument passed to `anova.lmm`. Right hand side can be specified via an attribute `"rhs"`.
`robust`	[logical] Should robust standard errors (aka sandwich estimator) be output instead of the model-based standard errors. Can also be `2` compute the degrees-of-freedom w.r.t. robust standard errors instead of w.r.t. model-based standard errors. Argument passed to `anova.lmm`.
`df`	[logical] Should the degrees-of-freedom be computed using a Satterthwaite approximation? Argument passed to `anova.lmm`.
`name.short`	[logical] use short names for the output coefficients (omit the name of the by variable).

`transform.sigma`, `transform.k`, `transform.rho`, `transform.names`
: [character] transformation used on certain type of parameters.

`trace` [interger, >0] Show the progress of the execution of the function.

Details

Grouping variable in argument repetition: when numeric, it will be converted into a factor variable, possibly adding a leading 0 to preserve the ordering. This transformation may cause inconsistency when combining results between different `lmm` object. This is why the grouping variable should preferably be of type character or factor.

See Also

`confint.mlmm` for a data.frame containing estimates with their uncertainty.
`summary.mlmm` for a summary of the model and estimates.
`autoplot.Wald_lmm` for a graphical display.

Examples

```
#### univariate regression ####
if(require(lava) && require(multcomp)){

set.seed(10)
d1 <- cbind(sim(lvm(Y~0.5*X1), 25), group = "A")
d2 <- cbind(sim(lvm(Y~0.1*X1), 100), group = "B")
d3 <- cbind(sim(lvm(Y~0.01*X1), 1000), group = "C")
d1$id <- 1:NROW(d1)
d2$id <- 1:NROW(d2)
d3$id <- 1:NROW(d3)

d <- rbind(d1,d2,d3)

e.mlmm <- mlmm(Y~X1, data = d, by = "group", effects = "X1=0")
summary(e.mlmm)
summary(e.mlmm, method = "single-step")
summary(e.mlmm, method = "single-step2")

## re-work contrast
summary(anova(e.mlmm, effects = mcp(X1 = "Dunnett")), method = "none")
## summary(mlmm(Y~X1, data = d, by = "group", effects = mcp(X1="Dunnett")))
}

#### multivariate regression ####
set.seed(10)
dL <- sampleRem(250, n.times = 3, format = "long")

e.mlmm <- mlmm(Y~X1+X2+X6, repetition = ~visit|id, data = dL,
               by = "X4", structure = "CS")
summary(e.mlmm)

e.mlmmX1 <- mlmm(Y~X1+X2+X6, repetition = ~visit|id, data = dL,
              by = "X4", effects = "X1=0", structure = "CS")
summary(e.mlmmX1)
summary(e.mlmmX1, method = "single-step")
```

model.frame.lmm *Extracting the Model Frame from a Linear Mixed Model*

Description

Contruct a data frame containing all the variables involved in a Linear Mixed Model.

Usage

```
## S3 method for class 'lmm'
model.frame(
  formula,
  newdata = NULL,
  type = NULL,
  add.index = FALSE,
  na.rm = TRUE,
  ...
)
```

Arguments

`formula`	a `lmm` object.
`newdata`	[data.frame] dataset relative to which the model frame should be constructed.
`type`	[character] by default returns the processed dataset used to fit the Linear Mixed Model (`NULL`). Can be used to add rows relative to missing repetitions (`"add.NA"`) or obtain a dataset with unique sets of covariates (`"unique"`) with respect to the mean structure.
`add.index`	[logical] should columns indexing the row number from the original dataset, time variable, cluster variable, strata variable be added to the output?
`na.rm`	[logical] should rows containing missing values for the variables used in the linear mixed model be removed? Not relevant when argument type is `"unique"`.
`...`	not used. For compatibility with the generic method.

Details

Column `"XXindexXX"` refers to the row of the original dataset (i.e. passed to argument `data` when calling `lmm`). When adding rows relative to missing repetitions, since there is no row in the original dataset, a negative sign is used.

Examples

```
data("armd.wide", package = "nlmeU")
e.lmH <- lmm(visual52 ~ lesion, structure = IND(~treat.f), data = armd.wide)
model.frame(e.lmH)
model.frame(e.lmH, add.index = TRUE)
model.frame(e.lmH, type = "unique")

data("gastricbypassL", package = "LMMstar")
dfL.NNA <- na.omit(gastricbypassL)
e.lmm <- lmm(glucagonAUC ~ time, repetition = ~visit|id, data = dfL.NNA, df = FALSE)
model.frame(e.lmm, type = "unique")
model.frame(e.lmm, type = c("unique","correlation"))
model.frame(e.lmm, type = "add.NA", add.index = TRUE)
```

---

model.matrix.lmm *Design Matrix for a Linear Mixed Model*

Description

Extract or construct design matrices for a linear mixed model.

Usage

```
## S3 method for class 'lmm'
model.matrix(
  object,
  newdata = NULL,
  effects = "mean",
  simplify = TRUE,
  drop.X = NULL,
  na.rm = TRUE,
  ...
)
```

Arguments

`object`	an `lmm` object
`newdata`	[data.frame] dataset relative to which the design matrix should be constructed.
`effects`	[character] design matrix relative to the mean model (`"mean"`), variance model (`"variance"`), correlation model (`"correlation"`), or all the previous (`"all"`). Can also be `"index"` to only output the normalize data and the cluster, time, strata indexes.
`simplify`	[logical] simplify the data format of the output (matrix instead of a list of matrix) when possible.
`drop.X`	[logical] when the design matrix does not have full rank, should columns be dropped?

`na.rm`	[logical] Should row containing missing values for the variables used in the linear mixed model be removed?
`...`	Not used. For compatibility with the generic method.

Value

When `simplify` is `FALSE`, a list with the followin elements:

- `mean`: design matrix for the mean model
- `Y`: vector of outcome values
- `vcov`: list of elements for the variance and correlation models.
- `index.cluster`: list containing, for each cluster, the location of the corresponding observations in the processed dataset.
- `index.clusterTime`: list containing, for each cluster, the repetition index corresponding observations.
- `index.clusterStrata`: list containing, for each cluster, the strata index corresponding observations.
- `param`: data.frame describing the modle parameters.
- `drop.X`: logical value indicating whether columns in the design matrix should be dropped if it has not full rank.
- `precompute.XX`, `precompute.XY`: moments of X and Y

When `simplify` is `TRUE`, this list will be simplified into a list with three elements:

- `mean`: design matrix for the mean model
- `variance`: design matrix for the variance model
- `correlation`: design matrix for the correlation model

or a single design matrixx.

`model.tables.lmm` *Statistical Inference and parametrization of a Linear Mixed Model*

Description

Export estimated parameters with their uncertainty (standard errors, degrees-of-freedom, confidence intervals and p-values) from a linear mixed model or a table describing each parameter (type, associated sigma or k parameter, ...).

Usage

```
## S3 method for class 'lmm'
model.tables(
  x,
  level = 0.95,
  effects = NULL,
  robust = FALSE,
  null = NULL,
  columns = NULL,
  df = NULL,
  type.information = NULL,
  transform.sigma = NULL,
  transform.k = NULL,
  transform.rho = NULL,
  transform.names = TRUE,
  backtransform = NULL,
  simplify = TRUE,
  ...
)
```

Arguments

`x` a `lmm` object.

`level` [numeric,0-1] the confidence level of the confidence intervals.

`effects` [character] Should the CIs/p-values for all coefficients be output (`"all"`), or only for mean coefficients (`"mean"` or `"fixed"`), or only for variance coefficients (`"variance"`), or only for correlation coefficients (`"correlation"`). Alternatively can be `"param"` to output the name and characteristics of each parameter (type, strata, ...).

`robust` [logical] Should robust standard errors (aka sandwich estimator) be output instead of the model-based standard errors. Can also be `2` compute the degrees-of-freedom w.r.t. robust standard errors instead of w.r.t. model-based standard errors.

`null` [numeric vector] the value of the null hypothesis relative to each coefficient.

`columns` [character vector] Columns to be output. Can be any of `"estimate"`, `"se"`, `"statistic"`, `"df"`, `"null"`, `"lower"`, `"upper"`, `"p.value"`.

`df` [logical] Should a Student's t-distribution be used to model the distribution of the coefficient. Otherwise a normal distribution is used.

`type.information`, `transform.sigma`, `transform.k`, `transform.rho`, `transform.names`
are passed to the `vcov` method. See details section in `coef.lmm`.

`backtransform` [logical] should the variance/covariance/correlation coefficient be backtransformed?

`simplify` [logical] omit from the output the backtransform attribute. Not relevant when the argument `effects="param"`,

`...` Not used. For compatibility with the generic method.

Details

When effects differs from "param", this function is a wrapper for confint with different default value for the argument column.

Value

A data.frame object.

model.tables.mlmm *Statistical Inference and parametrization of Multiple Linear Mixed Model*

Description

Combine estimated parameters with their uncertainty (standard errors, degrees-of-freedom, confidence intervals and p-values) from group-specific linear mixed models or a table describing each parameter (type, associated sigma or k parameter, ...).

Usage

```
## S3 method for class 'mlmm'
model.tables(
  x,
  parm = NULL,
  level = 0.95,
  method = NULL,
  df = NULL,
  columns = NULL,
  backtransform = NULL,
  ordering = "parameter",
  ...
)
```

Arguments

`x` a mlmm object.

`method` [character] type of adjustment for multiple comparisons, one of "none", "bonferroni", ..., "fdr", "single-step", "single-step2". and/or method(s) to pool the estimates, one of "average", "pool.se", "pool.gls", "pool.gls1", "pool.rubin". Only relevant when effects = "Wald".

`...` arguments to be passed to confint.lmm.

`effects` [character] Should the CIs/p-values for all coefficients be output ("all"), or only for mean coefficients ("mean" or "fixed"), or only for variance coefficients ("variance"), or only for correlation coefficients ("correlation"). Alternatively can be "param" to output the name and characteristics of each parameter (type, strata, ...).

`simplify` [logical] omit from the output the backtransform attribute. Not relevant when the argument effects="param",

Details

When `effects` is not `"param"`, this function simply calls `confint` with a specific value for the argument `column`.

Value

A `data.frame` object.

`model.tables.rbindWald_lmm` *Statistical Inference From Combined Wald Tests Applied to Linear Mixed Models*

Description

Combine estimates, standard errors, degrees-of-freedom, confidence intervals (CIs) and p-values relative to linear contrasts of parameters from different linear mixed models.

Usage

```
## S3 method for class 'rbindWald_lmm'
model.tables(
  x,
  effects = "Wald",
  level = 0.95,
  df = NULL,
  method = NULL,
  columns = NULL,
  ordering = NULL,
  backtransform = NULL,
  transform.names = TRUE,
  simplify = TRUE,
  ...
)
```

Arguments

`x` a `mlmm` object.

`effects` [character] should the linear contrasts involved in the Wald test be output (`"Wald"`), the contrast matrix (`"contrast"`), or the name/value/type of the underlying mixed model parameters (`"param"`)?

`level` [numeric, 0-1] nominal coverage of the confidence intervals.

`df` [logical] Should a Student's t-distribution be used to model the distribution of the Wald statistic. Otherwise a normal distribution is used.

`method` [character] Should pointwise confidence intervals be output (`"none"`) or simultaneous confidence intervals (`"bonferroni"`, ..., `"fdr"`, `"single-step"`, `"single-step2"`)?

`columns` [character vector] Columns to be output. Can be any of `"estimate"`, `"se"`, `"df"`, `"quantile"`, `"lower"`, `"upper"`, `"statistic"`, `"null"`, `"p.value"`.

`ordering` [character] should the output be ordered by name of the linear contrast (`"contrast"`) or by model (`"model"`).

`backtransform` [logical] should the estimates, standard errors, and confidence intervals be backtransformed?

`transform.names` [logical] should the name of the coefficients be updated to reflect the transformation that has been used? Only relevant when `effects="contrast"`.

`simplify` [logical] with argument `effects="Wald"`, omit from the output attributes containing additional information (e.g. approximation error made when adjusting p-values). with argument `effects="contrast"` the output will be converted into a matrix (instead of a list of matrix) whenever possible.

`...` Not used. For compatibility with the generic method.

`model.tables.Wald_lmm` *Statistical Inference for Wald tests Applied to a Linear Mixed Model*

Description

Export estimates, standard errors, degrees-of-freedom, confidence intervals (CIs) and p-values relative to linear contrasts of parameters from a linear mixed model.

Usage

```
## S3 method for class 'Wald_lmm'
model.tables(
  x,
  effects = "Wald",
  level = 0.95,
  df = NULL,
  method = NULL,
  columns = NULL,
  backtransform = NULL,
  transform.names = TRUE,
  simplify = TRUE,
  ...
)
```

Arguments

`x` a `Wald_lmm` object.

`effects` [character] should the linear contrasts involved in the Wald test be output (`"Wald"`), the contrast matrix (`"contrast"`), or the name/value/type of the underlying mixed model parameters (`"param"`)?

level [numeric, 0-1] nominal coverage of the confidence intervals.

df [logical] Should a Student's t-distribution be used to model the distribution of the Wald statistic. Otherwise a normal distribution is used.

method [character] Should pointwise confidence intervals be output ("none") or simultaneous confidence intervals ("bonferroni", ..., "fdr", "single-step", "single-step2")?

columns [character vector] Columns to be output. Can be any of "estimate", "se", "df", "quantile", "lower", "upper", "statistic", "null", "p.value".

backtransform [logical] should the estimates, standard errors, and confidence intervals be backtransformed?

transform.names
[logical] should the name of the coefficients be updated to reflect the transformation that has been used? Only relevant when effects="contrast".

simplify [logical] with argument effects="Wald", omit from the output attributes containing additional information (e.g. approximation error made when adjusting p-values). with argument effects="contrast" the output will be converted into a matrix (instead of a list of matrix) whenever possible.

... Not used. For compatibility with the generic method.

mt.test *Multiple Student's t-Test*

Description

Perform multiple Student's t-Test via heteroschedastic linear regression and combine the results, possibly adjusted for multiplicity.

Usage

```
mt.test(formula, data, method = NULL, level = 0.95, trace = TRUE)
```

Arguments

formula A formula like Y1+Y2+Y3~X|id with:

- the outcome on the left hand side (separated with +)
- the group variable on the right hand side
- a variable identifying each line in the dataset (optional)

data dataset in the wide format. Should inherit from data.frame.

method [character] type of adjustment for multiple comparisons, one of "none", "bonferroni", ..., "fdr", "single-step", "single-step2". See confint.Wald_lmm for more details. By default "single-step" when the test statistics have equal degrees-of-freedom and otherwise "single-step2".

level [numeric,0-1] the confidence level of the confidence intervals.

trace [logical] should a message be displayed in the console when there are missing data.

Details

In presence of missing values, performs a outcome specific complete case analysis.

Value

A data.frame with the estimates, confidence intervals, and p-values relative to each outcome. Depending on the argument `method` confidence intervals and p-values may be adjusted for multiple comparisons. The data.frame has an attribute `mlmm` containing the underlying regression models.

Examples

```
data(calciumW, package = "LMMstar")

t.test(bmd1 ~ grp, data = calciumW)

mt.test(bmd1+bmd2+bmd3+bmd4+bmd5 ~ grp, data = calciumW)
mt.test(bmd1+bmd2+bmd3+bmd4+bmd5 ~ grp|girl, data = calciumW)
mt.test(bmd1+bmd2+bmd3+bmd4+bmd5 ~ grp|girl, data = calciumW, method = "none")
```

`ncgsL` *Data From National Cooperative Gallstone Study (Long Format)*

Description

Data from the National Cooperative Gallstone Study (NCGS), a randomized study where the level of serum cholesterol was measured at baseline and after intake of high-dose chenondiol (750mg/day) or placebo. This dataset is in the long format (i.e. one line per measurement).

- `group`: treatment group (highdose or placebo).
- `id`: patient identifier.
- `visit`: visit index.
- `cholest`: cholesterol measurement.
- `time`: time after the start of the study at which the measurement has been done (in month). Treatment is given at 0+.

Usage

```
data(ncgsL)
```

References

Grundy SM, Lan SP, Lachin J. The effects of chenodiol on biliary lipids and their association with gallstone dissolution in the National Cooperative Gallstone Study (NCGS). J Clin Invest. 1984 Apr;73(4):1156-66. doi: 10.1172/JCI111301.

ncgsW *Data From National Cooperative Gallstone Study (Wide Format)*

Description

Data from the National Cooperative Gallstone Study (NCGS), a randomized study where the level of serum cholesterol was measured at baseline and after intake of high-dose chenondiol (750mg/day) or placebo. This dataset is in the wide format (i.e. one line per patient).

- `group`: treatment group (highdose or placebo).
- `id`: patient identifier.
- `cholest1`: cholesterol measurement at baseline (before treatment).
- `cholest2`: cholesterol measurement at 6 months (after treatment).
- `cholest3`: cholesterol measurement at 12 months (after treatment).
- `cholest4`: cholesterol measurement at 20 months (after treatment).
- `cholest5`: cholesterol measurement at 24 months (after treatment).

Usage

```
data(ncgsW)
```

References

Grundy SM, Lan SP, Lachin J. The effects of chenodiol on biliary lipids and their association with gallstone dissolution in the National Cooperative Gallstone Study (NCGS). J Clin Invest. 1984 Apr;73(4):1156-66. doi: 10.1172/JCI111301.

nobs.lmm *Number of Observations from a Linear Mixed Model*

Description

Extract the number of observations from a Linear Mixed Model.

Usage

```
## S3 method for class 'lmm'
nobs(object, ...)
```

Arguments

`object`	a `lmm` object.
`...`	Not used. For compatibility with the generic method.

Value

A vector with 4 elements:

- `obs`: the number of repetitions with full data
- `cluster`: the number of clusters with a least one repetition with full data
- `missing.obs`: the number of repetitions with missing data
- `missing.cluster`: the number of cluster with only missing data

`nobs.mlmm` *Number of Observations from Multiple Linear Mixed Models*

Description

Extract the number of observations from multiple linear mixed models.

Usage

```
## S3 method for class 'mlmm'
nobs(object, ...)
```

Arguments

`object`	a `mlmm` object.
`...`	Not used. For compatibility with the generic method.

Value

A matrix with as many rows as models and 4 columns:

- `obs`: the number of repetitions with full data
- `cluster`: the number of clusters with a least one repetition with full data
- `missing.obs`: the number of repetitions with missing data
- `missing.cluster`: the number of cluster with only missing data

onycholysisL *Data From The toenail onycholysis Study (Long Format)*

Description

Data from the toenail onychomycosis study, a randomized double-blind comparative study comparing Terbinafine (250 mg/day) against Itraconazole (200 mg/day) over 7 weeks. This dataset is in the long format (i.e. one line per measurement).

- `id`: patient identifier.
- `group`: treatment arm to which the patient has been randomized (`"itraconazole"` or `"terbinafine"`).
- `treatment`: treatment recieved by the patient at a given timepoint (`"none"`, `"itraconazole"`, or `"terbinafine"`).
- `visit`: index of time at which the measurement was taken (`1` to `7`).
- `time`: scheduled time, in months, for each measurement (`0`, `4`, `8`, `12`, `24`, `36`, `48`).
- `obstime`: time, in months, at which the measurement was taken (numeric between `0` and `18.5`).
- `response`: degree of onycholysis (separation of the nail plate from the nail-bed). Can be 0 for none or mild, 1 for moderate or severe.

Usage

```
data(onycholysisL)
```

References

De Backer et al. Twelve weeks of continuous oral therapy for toenail onychomycosis caused by dermatophytes: A double-blind comparative trial of terbinafine 250 mg/day versus itraconazole 200 mg/day, Journal of the American Academy of Dermatology, (2020) 110. doi: 10.1016/s0190-9622(98)70486-4

onycholysisW *Data From The toenail onycholysis Study (Wide Format)*

Description

Data from the toenail onychomycosis study, a randomized double-blind comparative study comparing Terbinafine (250 mg/day) against Itraconazole (200 mg/day) over 7 weeks. This dataset is in the wide format (i.e. one line per patient).

- `id`: patient identifier.
- `group`: treatment arm to which the patient has been randomized.
- `response1`,...,`response7`: degree of onycholysis (separation of the nail plate from the nail-bed). Can be 0 for none or mild, 1 for moderate or severe.
- `time1`,...,`time7`: time elapsed between baseline and the measurement (in months).

Usage

```
data(onycholysisW)
```

References

De Backer et al. Twelve weeks of continuous oral therapy for toenail onychomycosis caused by dermatophytes: A double-blind comparative trial of terbinafine 250 mg/day versus itraconazole 200 mg/day, Journal of the American Academy of Dermatology, (2020) 110. doi: 10.1016/s0190-9622(98)70486-4

---

partialCor *Partial Correlation*

Description

Estimate the partial correlation based on equation 19 of Lloyd et al 2008 (`partialCor.lmm`) or explicitely modeling the correlation via a linear mixed model (`partialCor.list`, `partialCor.formula`). The first option is numerically more efficient and exact with a single observation per cluster. With multiple repetitions, what is being estimated with the first option may not be clear and the second option is therefore preferrable.

Usage

```
partialCor(object, ...)

## S3 method for class 'list'
partialCor(
  object,
  data,
  repetition = NULL,
  structure = NULL,
  by = NULL,
  effects = NULL,
  rhs = NULL,
  method = "none",
  df = NULL,
  transform.rho = NULL,
  name.short = c(TRUE, FALSE),
  ...
)

## S3 method for class 'formula'
partialCor(object, repetition, ...)

## S3 method for class 'lmm'
partialCor(object, level = 0.95, R2 = FALSE, se = TRUE, df = TRUE, ...)
```

Arguments

`object` a formula with in the left hand side the variables for which the correlation should be computed and on the right hand side the adjustment set. Can also be a list of formula for outcome-specific adjustment set.

`...` arguments passed to `confint` for `partialCor.list` and `partialCor.formula`. Not used for `partialCor.lmm`.

`data` [data.frame] dataset containing the variables.

`repetition` [formula] Specify the structure of the data: the time/repetition variable and the grouping variable, e.g. ~ time|id.

`structure` [character] Specify the residual variance-covariance structure. Without repetitions, either `"UN"` or `"CS"`. With repetitions, one of `"UN"`, `"PEARSON"`, `"HLAG"`, `"LAG"`, `"HCS"`, `"CS"`.

`by` [character] variable used to stratified the correlation on.

`effects` [character or matrix] type of contrast to be used for comparing the correlation parameters. One of `"Dunnett"`, `"Tukey"`, `"Sequen"`, or a contrast matrix.

`rhs` [numeric vector] right hand side for the comparison of correlation parameters.

`method` [character] adjustment for multiple comparisons (e.g. `"single-step"`).

`df` [logical] Should a Student's t-distribution be used to model the distribution of the coefficient. Otherwise a normal distribution is used.

`transform.rho` [character] scale on which perform statistical inference (e.g. `"atanh"`)

`name.short` [logical vector of length 2] use short names for the output coefficients (omit the name of the by variable, omit name of the correlation parameter)

`level` [numeric,0-1] the confidence level of the confidence intervals.

`R2` [logical] Should the R2 (coefficient of determination) be computed?

`se` [logical] Should the uncertainty about the partial correlation be evaluated? Only relevant for `partialCor.lmm`.

Details

Fit a mixed model to estimate the partial correlation with the following variance-covariance pattern:

- **no repetition**: unstructure or compound symmetry structure for M observations, M being the number of variables on the left hand side (i.e. outcomes).
- **repetition**: structure for M*T observations where M being the number of variables (typically 2) and T the number of repetitions. Can be
  - `"UN"`: unstructured (except the off-diagonal containing the correlation parameter which is constant).
  - `"PEARSON"`: same as unstructured except it only uses a single variance parameter per variable, i.e. it assumes constant variance over repetitions.
  - `"HLAG"`: toeplitz by block with variable and repetition specific variance.
  - `"LAG"`: toeplitz by block, i.e. correlation depending on the gap between repetitions and specific to each variable. It assumes constant variance over repetitions.

  - "HCS": heteroschedastic compound symmetry by block, i.e. variable specific correlation constant over repetitions. A specific parameter is used for the off-diagonal crossing the variables at the same repetition (which is the marginal correlation parameter).
  - "CS": compound symmetry by block. It assumes constant variance and correlation over repetitions.

Value

A data.frame with the estimate partial correlation (rho), standard error, degrees-of-freedom, confidence interval, and p-value (test of no correlation). When `structure="CS"` or `structure="HCS"` is used with repeated measurements, a second correlation coefficient (r) is output where the between subject variance has been removed (similar to Bland et al. 1995).

References

Bland J M, Altman D G. Statistics notes: Calculating correlation coefficients with repeated observations: Part 1—correlation within subjects BMJ 1995; 310 :446 doi:10.1136/bmj.310.6977.446 Edwards, L.J., Muller, K.E., Wolfinger, R.D., Qaqish, B.F. and Schabenberger, O. (2008), An R2 statistic for fixed effects in the linear mixed model. Statist. Med., 27: 6137-6157. https://doi.org/10.1002/sim.3429

Examples

```
#### no repetition ####

## example from ppcor::pcor
y.data <- data.frame(
  hl=c(7,15,19,15,21,22,57,15,20,18),
  disp=c(0.000,0.964,0.000,0.000,0.921,0.000,0.000,1.006,0.000,1.011),
  deg=c(9,2,3,4,1,3,1,3,6,1),
  BC=c(1.78e-02,1.05e-06,1.37e-05,7.18e-03,0.00e+00,0.00e+00,0.00e+00
,  4.48e-03,2.10e-06,0.00e+00)
)
## ppcor::pcor(y.data)

## partial correlation based on a formula
partialCor(c(hl,disp)~BC+deg, data = y.data)
partialCor(hl + disp~BC+deg, data = y.data)
## partial correlation based on a list
partialCor(list(hl~BC+deg,disp~BC+deg), data = y.data)
## via an existing model
e.lm <- lmm(hl~disp+BC+deg, data = y.data)
partialCor(e.lm)

## using a different set of covariates for outcome
partialCor(list(hl~BC+deg, disp~BC), data = y.data)

## statified correlation (using another dataset)
data(gastricbypassW, package = "LMMstar")
gastricbypassW$weight.bin <- gastricbypassW$weight1>=120
partialCor(glucagonAUC1+glucagonAUC2~1, data = gastricbypassW, by = "weight.bin")
```

```
## compared correlation between groups
partialCor(glucagonAUC1+glucagonAUC2~1, data = gastricbypassW, by = "weight.bin",
           effects = "Dunnett")

#### with repetitions ####
## Not run:
data(gastricbypassL, package = "LMMstar")
## via a mixed model
eUN.lmm <- lmm(weight ~ glucagonAUC+time, repetition =~time|id,
               data = gastricbypassL, structure = "UN")
partialCor(eUN.lmm)

## mean: variable and timepoint specific mean parameter (8)
## variance: variable and timepoint specific variance parameter (8)
## correlation: correlation parameter specific for each variable and time lag (10)
e.cor <- partialCor(weight+glucagonAUC~time, repetition =~time|id,
                    data = gastricbypassL, structure = "LAG")
e.cor
coef(attr(e.cor,"lmm"), effects = "correlation")
if(require(ggplot2)){
autoplot(e.cor)
}

## same except for the mean structure: variable specific mean parameter (2)
e.cor2 <- partialCor(weight+glucagonAUC~time, repetition =~time|id,
                    data = gastricbypassL, structure = "LAG")

## mean: variable and timepoint specific mean parameter (8)
## variance: variable specific variance parameter (2)
## correlation: correlation parameter specific for each variable and some time lag (4)
e.cor3 <- partialCor(weight+glucagonAUC~time, repetition =~time|id,
                     data = gastricbypassL, structure = "CS")
e.cor3
coef(attr(e.cor3,"lmm"), effects = "correlation")
if(require(ggplot2)){
autoplot(e.cor3)
}


## End(Not run)
```

potassiumRepeatedL *Data From The Potassium Intake Study (Long Format with intermediate measurements)*

Description

Data from the potassium intake study, a randomized placebo-controlled crossover study where the effect of potassium supplement (90 mmol/day) on the renin-angiostensin-aldosteron system (RAAS) was assessed. This dataset is in the long format (i.e. one line per measurement) and contains measurement over 6 timepoints for each time period.

- `id`: patient identifier.
- `sequence`: treatment group to which the patient has been randomized.
- `period`: time period.
- `treatment`: treatment during the time period.
- `time`: time within each period.
- `aldo`: ??

Usage

```
data(potassiumRepeatedL)
```

References

Dreier et al. Effect of increased potassium intake on the reninangiotensinaldosterone system and subcutaneous resistance arteries: a randomized crossover study, Nephrol Dial Transplant (2020) 110. doi: 10.1093/ndt/gfaa114

`potassiumSingleL` *Data From The Potassium Intake Study (Long Format)*

Description

Data from the potassium intake study, a randomized placebo-controlled crossover study where the effect of potassium supplement (90 mmol/day) on the renin-angiostensin-aldosteron system (RAAS) was assessed. This dataset is in the long format (i.e. one line per measurement).

- `id`: patient identifier.
- `sequence`: treatment group to which the patient has been randomized.
- `period`: time period.
- `treatment`: treatment during the time period.
- `auc`: area under the curve of ?? during the time period.
- `bsauc`: ??
- `aldo`: ??

Usage

```
data(potassiumSingleL)
```

References

Dreier et al. Effect of increased potassium intake on the reninangiotensinaldosterone system and subcutaneous resistance arteries: a randomized crossover study, Nephrol Dial Transplant (2020) 110. doi: 10.1093/ndt/gfaa114

`potassiumSingleW` *Data From The Potassium Intake Study (Wide Format)*

Description

Data from the potassium intake study, a randomized placebo-controlled crossover study where the effect of potassium supplement (90 mmol/day) on the renin-angiostensin-aldosteron system (RAAS) was assessed. This dataset is in the wide format (i.e. one line per patient).

- `id`: patient identifier.
- `sequence`: treatment group to which the patient has been randomized.
- `treatment1`: treatment during the first time period.
- `treatment2`: treatment during the second time period.
- `auc1`: area under the curve of ?? during the first time period.
- `auc2`: area under the curve of ?? during the second time period.
- `bsauc1`: ??
- `aldo1`: ??
- `aldo2`: ??

Usage

```
data(potassiumSingleW)
```

References

Dreier et al. Effect of increased potassium intake on the reninangiotensinaldosterone system and subcutaneous resistance arteries: a randomized crossover study, Nephrol Dial Transplant (1998) 110. doi: 10.1093/ndt/gfaa114

`predict.lmm` *Predicted Outcome Value by a Linear Mixed Model.*

Description

Estimate the expected outcome conditional on covariates and possibly on other outcomes based on a linear mixed model.

Usage

```
## S3 method for class 'lmm'
predict(
  object,
  newdata,
  type = "static",
  p = NULL,
  se = NULL,
  robust = FALSE,
  df = NULL,
  level = 0.95,
  keep.data = NULL,
  format = "long",
  export.vcov = FALSE,
  simplify = TRUE,
  ...
)
```

Arguments

`object` a `lmm` object.

`newdata` [data.frame] a dataset containing covariate values to condition on. When setting the argument 'dynamic' predictions should also contain cluster, timepoint, and outcome values.

`type` [character] evaluate the expected outcome conditional on covariates only (`"static"`), the contribution of each variable to this 'static' prediction (`"terms"`), or the expected outcome conditional covariates and outcome values at other timepoints (`"dynamic"`). Based on the observed outcome and the 'dynamic' prediction for the missing outcome, can also evaluate the change from first repetitition (`"change"`), area under the curve (`"auc"`), and the area under the curve minus baseline (`"auc-b"`).

`p` [numeric vector] value of the model coefficients at which to evaluate the predictions. Only relevant if differs from the fitted values.

`se` [logical] should the standard error and confidence intervals for the predictions be output? It can also be a logical vector of length 2 to indicate the type of uncertainty to be accounted for: estimation and residual variance. In particular `c(TRUE,TRUE)` provides prediction intervals.

`robust` [logical] Should robust standard errors (aka sandwich estimator) be output instead of the model-based standard errors. Can also be `2` compute the degrees-of-freedom w.r.t. robust standard errors instead of w.r.t. model-based standard errors.

`df` [logical] should a Student's t-distribution be used to model the distribution of the predicted mean. Otherwise a normal distribution is used.

`level` [numeric,0-1] the confidence level of the confidence intervals.

`keep.data` [logical] should the dataset relative to which the predicted means are evaluated be output along side the predicted values? Only possible in the long format.

`format` [character] should the prediction be output in a matrix format with clusters in row and timepoints in columns (`"wide"`), or in a data.frame/vector with as many rows as observations (`"long"`)

`export.vcov` [logical] should the variance-covariance matrix of the prediction error be outcome as an attribute (`"vcov"`)?

`simplify` [logical] simplify the data format (vector instead of data.frame) and column names (no mention of the time variable) when possible.

`...` Not used. For compatibility with the generic method.

`vcov` [logical] should the variance-covariance matrix of the predictions be output as an attribute.

Details

Static prediction are made using the linear predictor $X\beta$ while dynamic prediction uses the conditional normal distribution of the missing outcome given the observed outcomes. So if outcome 1 is observed but not 2, prediction for outcome 2 is obtain by $X_2\beta + \sigma_{21}\sigma_{22}^{-1}(Y_1 - X_1\beta)$. In that case, the uncertainty is computed as the sum of the conditional variance $\sigma_{22} - \sigma_{21}\sigma_{22}^{-1}\sigma_{12}$ plus the uncertainty about the estimated conditional mean (obtained via delta method using numerical derivatives).

The model terms are computing similarly to `stats::predict.lm`, by centering the design matrix around the mean value of the covariates used to fit the model. Then the centered design matrix is multiplied by the mean coefficients and columns assigned to the same variable (e.g. three level factor variable) are summed together.

Value

When `format="long"`, a data.frame containing the following columns:

- `estimate`: predicted mean.
- `se`: uncertainty about the predicted mean.
- `df`: degrees-of-freedom
- `lower`: lower bound of the confidence interval of the predicted mean
- `upper`: upper bound of the confidence interval of the predicted mean

When `format="wide"`, a matrix containing the predict means (one line per cluster, one column per timepoint).

Examples

```
## simulate data in the long format
set.seed(10)
dL <- sampleRem(100, n.times = 3, format = "long")

## fit Linear Mixed Model
eUN.lmm <- lmm(Y ~ visit + X1 + X2 + X5,
               repetition = ~visit|id, structure = "UN", data = dL)

## prediction
```

```
newd <- data.frame(X1 = 1, X2 = 2, X5 = 3, visit = factor(1:3, levels = 1:3))
predict(eUN.lmm, newdata = newd)
predict(eUN.lmm, newdata = newd, keep.data = TRUE)
predict(eUN.lmm, newdata = newd, keep.data = TRUE, se = c(TRUE,TRUE))

## dynamic prediction
newd.d1 <- cbind(newd, Y = c(NA,NA,NA))
predict(eUN.lmm, newdata = newd.d1, keep.data = TRUE, type = "dynamic")
newd.d2 <- cbind(newd, Y = c(6.61,NA,NA))
predict(eUN.lmm, newdata = newd.d2, keep.data = TRUE, type = "dynamic")
newd.d3 <- cbind(newd, Y = c(1,NA,NA))
predict(eUN.lmm, newdata = newd.d3, keep.data = TRUE, type = "dynamic")
newd.d4 <- cbind(newd, Y = c(1,1,NA))
predict(eUN.lmm, newdata = newd.d4, keep.data = TRUE, type = "dynamic")
```

predict.mlmm *Predicted Outcome Value by Muliple Linear Mixed Model.*

Description

Estimate the expected outcome conditional on covariates and possibly on other outcomes based on group-specific linear mixed models.

Usage

```
## S3 method for class 'mlmm'
predict(
  object,
  p = NULL,
  newdata = NULL,
  keep.data = FALSE,
  simplify = TRUE,
  ...
)
```

profile.lmm *Log-Likelihood Contour For a Linear Mixed Model*

Description

Evaluate the (restricted) log-likelihood around Maximum Likelihood Estimate (MLE) of a linear mixed model. It will vary one parameter and either keep the other parameters constant or set the other parameters at their MLE (profile likelihood). Since a locally quadratic log-likelihood with an Hessian equivariant in law implies normally distributed estimates (Geyer 2013) it can help trusting confidence intervals and p-values in small samples with a non-normally distributed outcome.

Usage

```
## S3 method for class 'lmm'
profile(
  fitted,
  effects = NULL,
  profile.likelihood = FALSE,
  maxpts = NULL,
  level = 0.95,
  trace = FALSE,
  transform.sigma = NULL,
  transform.k = NULL,
  transform.rho = NULL,
  transform.names = TRUE,
  ...
)
```

Arguments

`fitted` a `lmm` object.

`effects` [character vector] name of the parameters who will be constrained. Alternatively can be the type of parameters, e.g. `"mean"`, `"variance"`, `"correlation"`, or `"all"`.

`profile.likelihood` [logical] should the other parameters be set to the value maximizing the constrained likelihood or stay at their MLE value?

`maxpts` [integer, >0] number of points use to discretize the likelihood, `maxpts` points smaller than the MLE and `maxpts` points higher than the MLE.

`level` [numeric, 0-1] the confidence level of the confidence intervals used to decide about the range of values for each parameter.

`trace` [logical] Show the progress of the execution of the function.

`transform.sigma` [character] Transformation used on the variance coefficient for the reference level. One of `"none"`, `"log"`, `"square"`, `"logsquare"` - see details.

`transform.k` [character] Transformation used on the variance coefficients relative to the other levels. One of `"none"`, `"log"`, `"square"`, `"logsquare"`, `"sd"`, `"logsd"`, `"var"`, `"logvar"` - see details.

`transform.rho` [character] Transformation used on the correlation coefficients. One of `"none"`, `"atanh"`, `"cov"` - see details.

`transform.names` [logical] Should the name of the coefficients be updated to reflect the transformation that has been used?

`...` Not used. For compatibility with the generic method.

Details

Each parameter defined by the argument `effets` is treated separately:

- the confidence interval of a parameter is discretized with `maxpt` points,
- this parameter is set to each discretization value.
- the other parameters are either set to the (unconstrained) MLE (`profile.likelihood=FALSE`) or to constrained MLE (`profile.likelihood=TRUE`). The latter case is much more computer intensive as it implies re-running the estimation procedure.
- the (restricted) log-likelihood is evaluated.

Value

A data.frame object containing the log-likelihood for various parameter values.

References

Geyer, C. J. (2013). Asymptotics of maximum likelihood without the lln or clt or sample size going to infinity. In Advances in Modern Statistical Theory and Applications: A Festschrift in honor of Morris L. Eaton, pages 1–24. Institute of Mathematical Statistics.

Examples

```
data(gastricbypassW, package = "LMMstar")
e.lmm <- lmm(weight2 ~ weight1 + glucagonAUC1,
             data = gastricbypassW, control = list(optimizer = "FS"))

## profile logLiklihood
## Not run:
e.pro <- profile(e.lmm, effects = "all", maxpts = 10, profile.likelihood = TRUE)
head(e.pro)
plot(e.pro)

## End(Not run)

## along a single parameter axis
e.sliceNone <- profile(e.lmm, effects = "all", maxpts = 10, transform.sigma = "none")
plot(e.sliceNone)
e.sliceLog <- profile(e.lmm, effects = "all", maxpts = 10, transform.sigma = "log")
plot(e.sliceLog)
```

`ranef.lmm` *Estimate Random Effect From a Linear Mixed Model*

Description

Recover the random effects from the variance-covariance parameters of a linear mixed model.

Usage

```
## S3 method for class 'lmm'
ranef(
  object,
  effects = "mean",
  scale = "absolute",
  se = FALSE,
  df = NULL,
  transform = (effects %in% c("std", "variance")),
  p = NULL,
  newdata = NULL,
  format = "long",
  simplify = TRUE,
  ...
)
```

Arguments

`object` a `lmm` object.

`effects` [character] should the estimated random effects (`"mean"`) or the estimated variance/standard deviation of the random effects (`"variance"`,`"std"`) be output?

`scale` [character] should the total variance, variance relative to each random effect, and residual variance be output (`"absolute"`). Or the ratio of these variances relative to the total variance (`"relative"`).

`se` [logical] should standard error and confidence intervals be evaluated using a delta method? Will slow down the execution of the function.

`df` [logical] Should degrees-of-freedom, computed using Satterthwaite approximation, be output.

`transform` [logical] should confidence intervals for the variance estimates (resp. relative variance estimates) be evaluated using a log-transform (resp. atanh transformation)?

`p` [numeric vector] value of the model coefficients to be used. Only relevant if differs from the fitted values.

`newdata` [data.frame] dataset relative to which the random effects should be computed. Only relevant if differs from the dataset used to fit the model.

`format` [character] should each type of random effect be output in a data.frame (`format="long"`)

`simplify` [logical] when relevant will convert list with a single element to vectors and omit unessential output.

`...` for internal use.

Details

Consider the following mixed model:

$$Y = X\beta + \epsilon = X\beta + Z\eta + \xi$$

where the variance of ϵ is denoted Ω, the variance of η is denoted Ω_η, and the variance of ξ is $\sigma^2 I$ with I is the identity matrix.
The random effets are estimating according to:

$$E[Y|\eta] = \Omega_\eta Z^t \Omega^{-1}(Y - X\beta)$$

Value

A data.frame or a list depending on the argument `format`.

Examples

```
if(require(nlme)){
data(gastricbypassL, package = "LMMstar")

## random intercept
e.RI <- lmm(weight ~ time + (1|id), data = gastricbypassL)
ranef(e.RI, effects = "mean")
ranef(e.RI, effects = "mean", se = TRUE)

ranef(e.RI, effects = "variance")
ranef(e.RI, effects = "variance", format = "wide")

}
```

`rbind.Wald_lmm` *Combine Wald Tests From Linear Mixed Models*

Description

Combine linear hypothesis tests from possibly different linear mixed models.

Usage

```
## S3 method for class 'Wald_lmm'
rbind(
  model,
  ...,
  effects = NULL,
  rhs = NULL,
  univariate = TRUE,
  multivariate = TRUE,
  name = NULL,
  name.short = TRUE,
  sep = ": "
)
```

Arguments

`model` a `Wald_lmm` object (output of `anova` applied to a `lmm` object)

`...` possibly other `Wald_lmm` objects

`effects` [character or numeric matrix] how to combine the left-hand side of the hypotheses. By default identity matrix but can also be `"Dunnett"`, `"Tukey"`, or `"Sequen"` (see function `multcomp::contrMat` from the multcomp package).

`rhs` [numeric vector] the right hand side of the hypothesis. Should have the same length as the number of row of argument `effects`.

`univariate` [logical] Should an estimate, standard error, confidence interval, and p-value be output for each hypothesis?

`multivariate` [logical] Should all hypotheses be simultaneously tested using a multivariate Wald test?

`name` [character vector or NULL] character used to identify each model in the output. By default, use the name of the outcome of the model.

`name.short` [logical] use short names for the output coefficients, e.g., omit the regression variable name when the same regression variable is used in all models.

`sep` [character] character used to separate the name/outcome and the covariate when identifying the linear hypotheses.

Details

In presence of measurements from the same cluster across several models, the influence function is used to estimate the covariance between the model parameters. This is why the (robust) standard errors may not match the (model-based) standard error from the linear mixed Setting the argument `robust` to `FALSE` when calling `anova.lmm` will rescale the (robust) standard errors to mimic the original model-based standard errors.

Examples

```
## simulate data
set.seed(10)
dL <- sampleRem(1e2, n.times = 3, format = "long")

## estimate mixed models
e.lmm1 <- lmm(Y ~ X1+X2+X3, repetition = ~visit|id, data = dL,
              structure = "CS", df = FALSE)
e.lmm2 <- lmm(Y ~ X1+X8+X9, repetition = ~visit|id, data = dL,
              structure = "CS", df = FALSE)


## combine null hypotheses
## - model-based standard errors
AAA <- anova(e.lmm1, effect = c("X1|X2,X3"="X1=0","X2|X1,X3"="X2=0"), simplify = FALSE)
BBB <- anova(e.lmm2, effect = c("X1|X8,X9"="X1=0"), simplify = FALSE)
ZZZ <- rbind(AAA,BBB)
summary(ZZZ) ## adjusted for multiple testing
rbind(model.tables(e.lmm1)[2:3,], model.tables(e.lmm2)[2,,drop=FALSE])
```

```
## select null hypotheses & combine (model-based like standard errors)
AA <- anova(e.lmm1, effect = c("X1|X2,X3"="X1=0","X2|X1,X3"="X2=0"),
            robust = TRUE, simplify = FALSE)
BB <- anova(e.lmm2, effect = c("X1|X8,X9"="X1=0"),
            robust = TRUE, simplify = FALSE)
ZZ <- rbind(AA,BB)
summary(ZZ)  ## adjusted for multiple testing
rbind(model.tables(e.lmm1, robust = TRUE)[2:3,],
      model.tables(e.lmm2, robust = TRUE)[2,,drop=FALSE])
```

RE *Random Effect Structure*

Description

Variance-covariance structure parametrized via random effects. Can be stratified on a categorical variable.

Usage

```
RE(formula, var.cluster, var.time, ranef = NULL, add.time)
```

Arguments

formula	formula indicating on which variable to stratify the residual variance and correlation (left hand side) and variables influencing the residual variance and correlation (right hand side).##'
var.cluster	[character] cluster variable.
var.time	[character] time variable.
ranef	[list] characteristics of the random effects
add.time	not used.

Details

A typical formula would be ~1, indicating a variance constant over time and the same correlation between all pairs of times.

Value

An object of class CS that can be passed to the argument structure of the lmm function.

Examples

```
RE(~1, var.cluster = "id", var.time = "time")
RE(~gender, var.cluster = "id", var.time = "time")
RE(gender~(1|id), var.time = "time")
```

rem *Remove Columns from Output*

Description

Auxiliary function that can be used when specifying the argument `columns` (e.g. calling `confint.lmm`) to remove columns. Not called remove to avoid confusion with `base::remove`

Usage

```
rem(...)
```

Arguments

`...`	[character vector] name of the columns to be removed to the default output.

Value

A character vector

Examples

```
set.seed(10)
dW <- sampleRem(25, n.times = 1, format = "long")
e.lmm <- lmm(Y~X1, data = dW)

confint(e.lmm, columns = rem("estimate"))
```

repetition *Number of Repetitions Within Cluster*

Description

Create a vector containing, for each line of the dataset, the number of occurences of the corresponding cluster up to the current line. Can stratify the number of occurences on one or several variables.

Usage

```
repetition(
  formula,
  data,
  format = "factor",
  keep.time = TRUE,
  sep = c(":", ".")
)
```

Arguments

`formula` [formula] Specify the structure of the data: the time/repetition variable and the grouping variable, e.g. ~1|id, ~ time|id, or ~time+region|id.

`data` [data.frame] dataset containing the observations.

`format` [character] the type of the output: a numeric vector (`"numeric"`), a character vector (`"character"`), or a factor vector (`"factor"`).

`keep.time` [logical] should the value of the time variable at the repetition be kept in the output (e.g. baseline.1, baseline.2, followUp.1 instead of 1,2,3). Only relevant when argument `formula` contain a time/repetition variable and `format="character"` or `format="factor"`.

`sep` [character vector of length 2] character strings used to combine time variables (first element) and the name of the time variable with its values (second element).

Value

A numeric/character/factor vector, depending on argument `format`.

Examples

```
data(sleepL, package = "LMMstar")
sleepL$dday <- paste0("d",sleepL$day)
sleepL$rep <- repetition(~1|id, data = sleepL)
sleepL$repDay <- repetition(~dday|id, data = sleepL)
sleepL$repDay.num <- repetition(~dday|id, data = sleepL, format = "numeric")
head(sleepL,15)
```

resample *Inference via Resampling for a Linear Mixed Model*

Description

Non-parametric bootstrap or permutation test for a linear mixed model.

Non-parametric bootstrap or permutation test for group-specific linear mixed models.

Usage

```
resample(object, type, ...)

## S3 method for class 'lmm'
resample(
  object,
  type,
  effects,
  n.sample = 1000,
  studentized = TRUE,
  trace = TRUE,
  seed = NULL,
  cpus = 1,
  export.cpus = NULL,
  ...
)

## S3 method for class 'mlmm'
resample(
  object,
  type,
  method = NULL,
  n.sample = 1000,
  studentized = TRUE,
  trace = TRUE,
  seed = NULL,
  cpus = 1,
  export.cpus = NULL,
  ...
)
```

Arguments

`object` a `mlmm` object.

`type` [character] should permutation test (`"perm-var"` or `"perm-res"`) or non-parametric bootstrap (`"boot"`) be used?

`...` Not used. For compatibility with the generic method.

`effects` [character vector] the variable(s) to be permuted or the effect(s) to be tested via non-parametric bootstrap. Can also be a function of the model parameters when performing non-parametric bootstrap.

`n.sample` [integer] the number of samples used.

`studentized` [logical] should a studentized boostrap or permutation test be used?

`trace` [logical] should the execution of the resampling be traced?

`seed` [integer, >0] Random number generator (RNG) state used when starting resampling.

`cpus` [integer, >0] number of child-processes for parallel evaluation. If `NULL` no state is set.

`export.cpus` [character vector] name of the variables to export to each cluster.

`method` [character] type of adjustment for multiple comparisons, one of `"none"`, `"bonferroni"`, ..., `"fdr"`, `"single-step"`, `"single-step2"`. and/or method(s) to pool the estimates, one of `"average"`, `"pool.se"`, `"pool.gls"`, `"pool.gls1"`, `"pool.rubin"`.

Details

All approach are carried at the cluster level:

- Bootstrap: sampling with replacement clusters. If a cluster is picked twice then different cluster id is used for each pick.
- Permutation: permuting covariate values between clusters (this only lead to the null hypothesis when the covariate values are constant within clusters) or assigning new outcome values by, under the null, permuting the normalized residuals, rescaling to residuals, and adding back the permuted fixed effect (any mean effect under H1 would be 0 because of the permutation if the variance-covariance structure is correct). The later procedure originates from Oliver et al (2012).

The studentized bootstrap keep the original estimate and standard error but uses the samples to evaluates the quantiles of the distribution used to form the confidence intervals. The studentized permutation test approximate the distribution of the test statistic under the null (instead of the distribution of the estimate under the null.).

P-values for the bootstrap are computed by centering the bootstrap distribution of the estimate or test statistic around 0 and evaluating the frequency at which it takes values more extreme than the observed estimate or test statistics.

References

Oliver E. Lee and Thomas M. Braun (2012), **Permutation Tests for Random Effects in Linear Mixed Models**. *Biometrics*, Journal 68(2).

Examples

```
## Not run:

#### univariate linear regression ####
data(gastricbypassW, package = "LMMstar")
## rescale to ease optimization
gastricbypassW$weight1 <- scale(gastricbypassW$weight1)
gastricbypassW$weight2 <- scale(gastricbypassW$weight2)
gastricbypassW$glucagonAUC1 <- scale(gastricbypassW$glucagonAUC1)

e.lm <- lmm(weight2~weight1+glucagonAUC1, data = gastricbypassW)
model.tables(e.lm)

## non-parametric bootstrap
resample(e.lm, type = "boot", effects = c("weight1","glucagonAUC1"), seed = 10)
## permutation test
resample(e.lm, type = "perm-var", effects = "weight1", seed = 10)
resample(e.lm, type = "perm-var", effects = "glucagonAUC1", seed = 10)
## using multiple cores
resample(e.lm, type = "boot", effects = c("weight1","glucagonAUC1"), cpus = 4)

#### random intercept model ####
data(gastricbypassL, package = "LMMstar")
gastricbypassL$weight <- scale(gastricbypassL$weight)
gastricbypassL$glucagonAUC <- scale(gastricbypassL$glucagonAUC)
gastricbypassL$gender <- as.numeric(gastricbypassL$id) %% 2
gastricbypassLR <- na.omit(gastricbypassL)
```

```
eCS.lmm <- lmm(weight~glucagonAUC+gender, data = gastricbypassLR,
               repetition = ~visit|id, structure = "CS")
model.tables(eCS.lmm)

## non-parametric bootstrap
resample(eCS.lmm, type = "boot", effects = c("glucagonAUC","gender"), seed = 10, trace = FALSE)
## permutation test
resample(eCS.lmm, type = "perm-var", effects = "gender", seed = 10)
resample(eCS.lmm, type = "perm-res", effects = "glucagonAUC", seed = 10)

## End(Not run)
```

residuals *Extract The Residuals From a Linear Mixed Model*

Description

Extract or compute the residuals of a linear mixed model.

Usage

```
## S3 method for class 'lmm'
residuals(
  object,
  type = "response",
  variable = NULL,
  at = NULL,
  newdata = NULL,
  p = NULL,
  format = "long",
  keep.data = FALSE,
  fitted.ci = FALSE,
  simplify = TRUE,
  var,
  ...
)

## S3 method for class 'clmm'
residuals(object, ...)

## S3 method for class 'mlmm'
residuals(
  object,
  p = NULL,
  newdata = NULL,
  keep.data = FALSE,
  simplify = TRUE,
    ...
)
```

Arguments

`object` a `lmm` object.

`type` [character] type of residual to output such as raw residuals (`"response"`), normalized residuals (`"normalized"`, partial residuals (`"partial"`). Can also be `"all"` to output all except partial residuals. See detail section.

`variable` [character vector] name of the variable relative to which the partial residuals should be computed.

`at` [data.frame] values for the covariates at which to evaluate the partial residuals.

`newdata` [data.frame] dataset relative to which the residuals should be computed. Only relevant if differs from the dataset used to fit the model.

`p` [numeric vector] value of the model coefficients at which to evaluate the residuals. Only relevant if differs from the fitted values.

`format` [character] Should the residuals be output in a matrix format with clusters in row and timepoints in columns (`"wide"`), or in a data.frame/vector with as many rows as observations (`"long"`)

`keep.data` [logical] Should the dataset relative to which the residuals are evaluated be output along side the residual values? Only possible in the long format.

`fitted.ci` [logical] Should the confidence intervals relative to the fitted values be added to the output. Only relevant when argument `keep.data=TRUE`.

`simplify` [logical] Simplify the data format (vector instead of data.frame) and column names (no mention of the time variable) when possible. Otherwise, information about the call and reference values used for partial residuals be added as an attribute.

`var` [Deprecated] Not used anymore, replaced by argument variable.

`...` Not used. For compatibility with the generic method.

Details

The argument `type` defines how the residuals are computed:

- `"response"`: raw residual, i.e. observed outcome minus fitted value $\varepsilon_{ij} = Y_{ij} - X_{ij}\hat{\beta}$.
- `"pearson"`: each raw residual is divided by its modeled standard deviation $\varepsilon_{ij} = \frac{Y_{ij}-X_{ij}\hat{\beta}}{\sqrt{\hat{\omega}_{ij}}}$.
- `"studentized"`: same as `"pearson"` but excluding the contribution of the cluster in the modeled standard deviation $\varepsilon_{ij} = \frac{Y_{ij}-X_{ij}\hat{\beta}}{\sqrt{\hat{\omega}_{ij}-\hat{q}_{ij}}}$.
- `"normalized"`: raw residuals are multiplied, within clusters, by the inverse of the (upper) Cholesky factor of the modeled residual variance covariance matrix $\varepsilon_{ij} = (Y_i - X_i\hat{\beta})\hat{C}^{-1}$.
- `"normalized2"`: raw residuals are multiplied, within clusters, by the inverse of the modeled residual variance covariance matrix $\varepsilon_{ij} = (Y_i - X_i\hat{\beta})O\hat{mega}^{-1}$.

- "scaled": scaled residuals (see PROC MIXED in SAS). Numerically identical to "normalized" but computed by sequentially scaling and centering the residuals, to make them conditionally independent of previous residuals from the same cluster at previous repetitions.
- "partial": partial residuals ($\gamma E + \hat{\varepsilon}$). A reference level can be also be specified via the attribute "reference" to change the absolute level of the partial residuals. "partial-center": partial residuals with centered continuous covariates ($\gamma E + \hat{\varepsilon}$ where E has been centered, i.e., has 0-mean)

where

- $X = (E, W)$ the design matrix. For partial residuals, it is split according to the variable(s) in argument variable (E) and the rest (W).
- Y the outcome
- $\hat{\beta} = (\hat{\gamma}, \hat{\delta})$ the estimated mean coefficients relative to $X = (E, W)$
- $\hat{\Omega}$ the modeled variance-covariance of the residuals and $\hat{\omega}$ its diagonal elements
- $\hat{C}$ the upper Cholesky factor of $\hat{\Omega}$, i.e. upper triangular matrix satisfying $\hat{C}^t\hat{C} = \hat{\Omega}$
- $\hat{Q}_i = X_i(X^t\hat{\Omega}X)^{-1}X_i^t$ a cluster specific correction factor, approximating the contribution of cluster i to $\hat{\Omega}$. Its diagonal elements are denoted $\hat{q}_i$.
- $\hat{D}_i$ the upper Cholesky factor of $\hat{\Omega} - \hat{Q}_i$

Setting argument fitted.ci to TRUE, simplify to FALSE, format to "long" returns an attribute "grad" containing the first order partial derivatives of the residuals with respect to the model parameters.

Value

lmm: a vector or a data.frame when format="long" (one line per observation, one column per type of residual), a matrix when format="wide" (one line per cluster, one column per timepoint).

Examples

```
#### simulate data in the long format ####
set.seed(10)
dL <- sampleRem(100, n.times = 3, format = "long")

#### Linear Model ####
e.lm <- lmm(Y ~ visit + X1 + X2 + X6, data = dL)

## partial residuals
pRes <- residuals(e.lm, type = "partial", variable = "X6")
range(residuals(e.lm) + dL$X6 * coef(e.lm)["X6"] - pRes)
e.reslm <- residuals(e.lm, type = "partial", variable = "X6", keep.data = TRUE, simplify = FALSE)
plot(e.reslm)

## partial residuals with specific reference
residuals(e.lm, type = "partial", variable = "X1",
          at = data.frame(visit=factor(2,1:3),X2=0,X6=3))

## partial residuals with centered covariates
```

```
residuals(e.lm, type = "partial-center", variable = "X1")

#### Linear Mixed Model ####
eUN.lmm <- lmm(Y ~ visit + X1 + X2 + X5 + X6,
               repetition = ~visit|id, structure = "UN", data = dL)

## residuals
e.resL <- residuals(eUN.lmm, type = "normalized",
                    keep.data = TRUE, simplify = FALSE)
plot(e.resL, type = "qqplot")
plot(e.resL, type = "scatterplot", labeller = ggplot2::label_both)
e.resW <- residuals(eUN.lmm, format = "wide", type = "normalized",
                    simplify = FALSE)
plot(e.resW, type = "correlation")

## residuals and predicted values
residuals(eUN.lmm, type = "all")
residuals(eUN.lmm, type = "all", keep.data = TRUE)

## partial residuals
residuals(eUN.lmm, type = "partial", variable = c("(Intercept)","visit","X6"))
```

sampleRem *Sample Longuitudinal Data*

Description

Sample longuitudinal data with covariates

Usage

```
sampleRem(
  n,
  n.times,
  mu = 1:n.times,
  sigma = rep(1, n.times),
  lambda = rep(1, n.times),
  beta = c(2, 1, 0, 0, 0, 1, 1, 0, 0, 0),
  gamma = matrix(0, nrow = n.times, ncol = 10),
  format = "wide",
  latent = FALSE
)
```

Arguments

`n` [integer,>0] sample size

`n.times` [integer,>0] number of visits (i.e. measurements per individual).

`mu` [numeric vector] expected measurement value at each visit (when all covariates are fixed to 0). Must have length `n.times`.

`sigma` [numeric vector,>0] standard error of the measurements at each visit (when all covariates are fixed to 0). Must have length `n.times`.

`lambda` [numeric vector] covariance between the measurement at each visit and the individual latent variable. Must have length `n.times`.

`beta` [numeric vector of length 10] regression coefficient between the covariates and the latent variable.

`gamma` [numeric matrix with n.times rows and 10 columns] regression coefficient specific to each timepoint (i.e. interaction with time).

`format` [character] Return the data in the wide format (`"wide"`) or long format (`"long"`). Can also be `"wide+"` or `"long+"` to export as attributes the function arguments and the latent variable model used to generate the data.

`latent` [logical] Should the latent variable be output?

Details

The generative model is a latent variable model where each outcome (Y_j) load on the latent variable (η) with a coefficient lambda:

$$Y_j = \mu_j + \lambda_j * \eta + \sigma_j \epsilon_j$$

The latent variable is related to the covariates ($X_1, \ldots, X_(10)$):

$$\eta = \alpha + \beta_1 X_1 + ... + \beta_{10} X_{10} + \xi$$

ϵ_j and ξ are independent random variables with standard normal distribution.

Value

a data.frame

Examples

```
#### generate data in the wide format ####
set.seed(10)
dW <- sampleRem(100, n.times = 3, format = "wide")
head(dW)

#### generate data in the long format ####
set.seed(10)
dL <- sampleRem(100, n.times = 3, format = "long")
head(dL)
```

scatterplot *Scatterplot for Continuous Variables*

Description

Produce a matrix of plot for continuous variables: scatterplots, histograms, correlation and missing values. Inspired from the ggpairs function of the R package GGally.

Usage

```
scatterplot(
  data,
  formula,
  columns,
  format = NULL,
  group = NULL,
  transform = NULL,
  facet = "grid_both",
  alpha.point = 1,
  type.diag = "histogram",
  bins = NULL,
  position.bar = "identity",
  linewidth.density = NULL,
  alpha.area = NULL,
  method.cor = "pearson",
  name.cor = "r",
  size.cor = NULL,
  digits = c(3, 2),
  display.NA = NULL,
  color = NULL,
  xlim = NULL,
  ylim = NULL,
  size.axis = NULL,
  size.legend = NULL,
  size.facet = NULL
)
```

Arguments

data	[data.frame] dataset containing the variables to be displayed.
formula	[formula] formula indicating the variables to be used (outcome~time|id). Long format only.
columns	[character vector] Columns whose numerical values are to be displayed. Wide format only.
format	[character] Is the dataset in the long ("long") or wide ("wide") format?

`group` [character] optional group variable used to color the points, stratify the histogram/density and correlation.

`transform` [character or function] optional transformation to be applied on the outcome.

`facet` [character] whether to use `ggplot:::facet_grid` (`"grid"`) or `ggh4x::facet_grid2` (`"grid2"`).

`alpha.point` [numeric] the transparency level used to display the points in the scatterplot.

`type.diag` [character] type of graphical display on the diagonal: `"boxplot"`, `"histogram"`, or `"density"`.

`bins` [character or numeric vector] algorithm or values or number of values used to create the histogram cells. When using `facet="grid2"` and `density=TRUE` a character of length two indicating the bandwith and the kernel to be used. See `ggplot2::stat_density`.

`position.bar` [character] passed to `geom_histogram` (argument `position`). Only relevant when having multiple groups and using `ggh4x::facet_grid2`.

`linewidth.density`
[numeric,>0] width of the lines on the density plot.

`alpha.area` [numeric, 0-1] the transparency level used to display the area under the density curve or histogram.

`method.cor` [character] estimator of the correlation. Argument passed to `stats::cor`. When `NA`, the correlation is not displayed.

`name.cor` [character] character used to represent the correlation. By default `"r"` but can be changed to `"\u03C1"` to display the greek letter ρ.

`size.cor` [numeric,>0] size of the font used to display the correlation or information about missing values.

`digits` [numeric of length 2] number of digits used to display the correlation or round the percentage of missing values.

`display.NA` [0:2 or "only"] Should the number of missing values be displayed. When taking value 2, will also display the percentage of missing values.

`color` [character vector] color used to display the values for each group.

`xlim` [numeric,>0 or "common"] range of the x-axis.

`ylim` [numeric,>0 or "common"] range of the y-axis.

`size.axis` [numeric,>0] size of the font used to display the tick labels.

`size.legend` [numeric,>0] size of the font used to display the legend. When argument `facet="grid"`, can have a second element to control the size of the legend key and a third to control the width of the legend.

`size.facet` [numeric,>0] size of the font used to display the facets (row and column names). When argument `facet="grid"`, can have a second and third elements to control the width and height of the facets.

Details

In the long format, the outcome variable contains the numerical values to be displayed. The time variable will be used to spit outcome and display each split separately or jointly with one other split. The identifier links the outcome values across time.

Value

a list of ggplot objects (`facet="grid"`) or a ggplot object (`facet="grid2"`)

Examples

```
data(gastricbypassL, package = "LMMstar")
gastricbypassL$group <- as.numeric(gastricbypassL$id) %% 3
data(gastricbypassW, package = "LMMstar")

## single group (wide or long format)
scatterplot(gastricbypassL, formula = weight~time|id)
scatterplot(gastricbypassW, columns = paste0("weight",1:4))

## Not run:
## change the number of bins
scatterplot(gastricbypassL, formula = weight~time|id, type.diag = "hist", bins = 15)

## remove variable name in facet
scatterplot(gastricbypassL, formula = weight~time|id, facet = "grid")

## use boxplot instead of histogram
scatterplot(gastricbypassL, formula = weight~time|id, type.diag = "boxplot")

## same scale
scatterplot(gastricbypassL, formula = weight~time|id,
            xlim = "common", ylim = "common")

## transform outcome
scatterplot(gastricbypassL, formula = weight~time|id, transform = "log")

## handling missing values
scatterplot(gastricbypassL, formula = glucagonAUC~time|id)

## coloring per group
scatterplot(gastricbypassL, formula = weight~time|id, group = "group")

## only display NAs
scatterplot(gastricbypassL, formula = glucagonAUC~time|id,
            display.NA = "only", group = "group")

## increase size of the legend (text, symbol, width)
scatterplot(gastricbypassL, formula = glucagonAUC~time|id,
            display.NA = "only", group = "group", size.legend = c(20,3,0.5))

## End(Not run)
```

schoolL *Simulated Data with 3-level struture (Long Format)*

Description

Simulated data a nested structure: Student/Class/School and one outcome.

- `school`:
- `class`:
- `student`:
- `outcome`:

Usage

```
data(schoolL)
```

`score.lmm` *Extract The Score From a Linear Mixed Model*

Description

Extract or compute the first derivative of the log-likelihood of a linear mixed model.

Usage

```
## S3 method for class 'lmm'
score(
  x,
  effects = "mean",
  indiv = FALSE,
  newdata = NULL,
  p = NULL,
  transform.sigma = NULL,
  transform.k = NULL,
  transform.rho = NULL,
  transform.names = TRUE,
  ...
)
```

Arguments

`x` a `lmm` object.

`effects` [character] Should the score relative to all coefficients be output (`"all"`),

`indiv` [logical] Should the contribution of each cluster to the score be output? Otherwise output the sum of all clusters of the derivatives. or only coefficients relative to the mean (`"mean"` or `"fixed"`), or only coefficients relative to the variance and correlation structure (`"variance"` or `"correlation"`).

`newdata` [data.frame] dataset relative to which the score should be computed. Only relevant if differs from the dataset used to fit the model.

`p` [numeric vector] value of the model coefficients at which to evaluate the score. Only relevant if differs from the fitted values.

`transform.sigma` [character] Transformation used on the variance coefficient for the reference level. One of `"none"`, `"log"`, `"square"`, `"logsquare"` - see details.

`transform.k` [character] Transformation used on the variance coefficients relative to the other levels. One of `"none"`, `"log"`, `"square"`, `"logsquare"`, `"sd"`, `"logsd"`, `"var"`, `"logvar"` - see details.

`transform.rho` [character] Transformation used on the correlation coefficients. One of `"none"`, `"atanh"`, `"cov"` - see details.

`transform.names` [logical] Should the name of the coefficients be updated to reflect the transformation that has been used?

`...` Not used. For compatibility with the generic method.

Details

For details about the arguments **transform.sigma**, **transform.k**, **transform.rho**, see the documentation of the coef.lmm function.

Value

When argument indiv is `FALSE`, a vector with the value of the score relative to each coefficient. When argument indiv is `TRUE`, a matrix with the value of the score relative to each coefficient (in columns) and each cluster (in rows).

`score.mlmm` *Extract The Score From Multiple Linear Mixed Models*

Description

Extract or compute the first derivative of the log-likelihood of each linear mixed model.

Usage

```
## S3 method for class 'mlmm'
score(
  x,
  effects = "contrast",
  indiv = FALSE,
  p = NULL,
  newdata = NULL,
  ordering = "by",
  simplify = TRUE,
  ...
)
```

Arguments

x	a mlmm object.
effects	[character] By default will output the estimates relative to the hypotheses being tested ("contrast"). But can also output all model coefficients ("all"), or only coefficients relative to the mean ("mean" or "fixed"), or only coefficients relative to the variance structure ("variance"), or only coefficients relative to the correlation structure ("correlation").
indiv	[logical] Should the contribution of each cluster to the score be output? Otherwise output the sum of all clusters of the derivatives.
p	[list of numeric vector] list of model coefficients to be used. Only relevant if differs from the fitted values.
newdata	[NULL] Not used. For compatibility with the generic method.
ordering	[character] should the output be ordered by type of parameter (parameter) or by model (by).
simplify	[logical] should the score be combined across models into a single vector (indiv=FALSE) or matrix (indiv=TRUE)?
...	passed to score.lmm.

Value

When argument indiv is FALSE, a vector with the value of the score relative to each coefficient. When argument indiv is TRUE, a matrix with the value of the score relative to each coefficient (in columns) and each cluster (in rows). When effects differs from "contrast" and simplify=FALSE, it will store the score in a list with an element relative to each parameter or model (argument ordering).

sigma.lmm *Extract The Residuals Variance-Covariance Matrix From a Linear Mixed Model*

Description

Extract the unique set of residuals variance-covariance matrices or the one relative to specific clusters.

Usage

```
## S3 method for class 'lmm'
sigma(
  object,
  cluster = NULL,
  p = NULL,
  chol = FALSE,
  inverse = FALSE,
  simplify = TRUE,
  ...
)
```

Arguments

`object` a `lmm` object.

`cluster` [character, data.frame, NULL] identifier of the cluster(s) for which to extract the residual variance-covariance matrix. For new clusters, a dataset containing the information (cluster, time, strata, ...) to be used to generate the residual variance-covariance matrices. When `NULL`, will output complete data covariance patterns.

`p` [numeric vector] value of the model coefficients at which to evaluate the residual variance-covariance matrix. Only relevant if differs from the fitted values.

`chol` [logical] Output the cholesky factorization of the variance-covariance matrix.

`inverse` [logical] Output the matrix inverse of the variance-covariance matrix.

`simplify` [logical] When there is only one variance-covariance matrix, output a matrix instead of a list of matrices.

`...` Not used. For compatibility with the generic method.

Value

A list where each element contains a residual variance-covariance matrix. Can also be directly a matrix when argument is `simplify=TRUE` and there is a single residual variance-covariance matrix.

Examples

```
## simulate data in the long format
set.seed(10)
dL <- sampleRem(100, n.times = 3, format = ”long”)
dL$id.fac <- paste0(”id”,dL$id)

## fit Linear Mixed Model
eUN.lmm <- lmm(Y ~ X1 + X2 + X5, repetition = ~visit|id.fac,
               structure = ”UN”, data = dL, df = FALSE)

## extract residuals variance covariance matrix
sigma(eUN.lmm) ## unique patterns
sigma(eUN.lmm, cluster = c(”id1”,”id5”)) ## existing clusters
sigma(eUN.lmm, cluster = dL[1:7,,drop=FALSE]) ## new clusters
```

summarize *Summary Statistics*

Description

Compute summary statistics for multiple variables and/or multiple groups and save them in a data frame.

Usage

```
summarize(
  formula,
  data,
  repetition = NULL,
  columns = NULL,
  FUN = NULL,
  na.action = stats::na.pass,
  na.rm = FALSE,
  level = 0.95,
  skip.reference = FALSE,
  digits = NULL,
  filter = NULL,
  ...
)
```

Arguments

`formula` [formula] on the left hand side the outcome(s) and on the right hand side the grouping variables. E.g. Y1+Y2 ~ Gender + Gene will compute for each gender and gene the summary statistics for Y1 and for Y2.

`data` [data.frame] dataset containing the observations.

`repetition` [formula] Specify the structure of the data: the time/repetition variable and the grouping variable, e.g. ~ time|id. Used in the long format to count the number of missing values (i.e. add the number of missing rows) and evaluate the correlation.

`columns` [character vector] name of the summary statistics to kept in the output. Can be any of, or a combination of:

- `"observed"`: number of observations with a measurement.
- `"missing"`: number of missing observations. When specifying a grouping variable, it will also attempt to count missing rows in the dataset.
- `"pc.missing"`: percentage missing observations.
- `"mean"`, `"mean.lower"` `"mean.upper"`: mean with its confidence interval.
- `"median"`, `"median.lower"` `"median.upper"`: median with its confidence interval.
- `"sd"`, `"sd.lower"`, `"sd.upper"`: standard deviation around the mean with its confidence interval.
- `"sd0"`, `"sd0.lower"`, `"sd0.upper"`: standard deviation around 0 with its confidence interval.
- `"skewness"`: skewness, as the third standardized moment.
- `"kurtosis"`: kurtosis, as the fourth standardized moment.
- `"q1"`, `"q3"`, `"IQR"`: 1st and 3rd quartile, interquartile range.
- `"min"`, `"max"`: minimum and maximum observation.
- `"predict.lower"`, `"predict.upper"`: prediction interval for normally distributed outcome.

- "correlation": correlation matrix between the outcomes (when feasible, see detail section).

FUN
: [function] user-defined function for computing summary statistics. It should take a vector as an argument and output a named single value or a named vector.

na.action
: [function] a function which indicates what should happen when the data contain 'NA' values. Passed to the stats::aggregate function.

na.rm
: [logical] Should the summary statistics be computed by omitting the missing values.

level
: [numeric,0-1] the confidence level of the confidence intervals.

skip.reference
: [logical] should the summary statistics for the reference level of categorical variables be omitted?

digits
: [integer, >=0] the minimum number of significant digits to be used to display the results. Passed to print.data.frame

filter
: [character] a regular expression passed to grep to filter the columns of the dataset. Relevant when using . to indicate all other variables.

...
: additional arguments passed to argument FUN.

Details

This function is essentially an interface to the stats::aggregate function.
WARNING: it has the same name as a function from the dplyr package. If you have loaded dplyr already, you should use ::: to call summarize i.e. use LMMstar:::summarize.

Confidence intervals (CI) and prediction intervals (PI) for the mean are computed via stats::t.test. Confidence intervals (CI) for the standard deviation are computed using a chi-squared approximation. Confidence intervals (CI) for the median are computed via asht::medianTest.

Value

A data frame containing summary statistics (in columns) for each outcome and value of the grouping variables (rows). It has an attribute "correlation" when it was possible to compute the correlation matrix for each outcome with respect to the grouping variable.

See Also

correlate for correlation matrix.

Examples

```
#### simulate data (wide format) ####
set.seed(10)
d <- sampleRem(1e2, n.times = 3)
d$treat <-  sample(LETTERS[1:3], NROW(d), replace=TRUE, prob=c(0.3, 0.3, 0.4) )

## add a missing value
d2 <- d
d2[1,"Y2"] <- NA
```

```
#### summarize (wide format) ####

## summary statistic (single variable)
summarize(Y1 ~ 1, data = d)
## stratified summary statistic (single variable)
summarize(Y1 ~ X1, data = d2)
## stratified summary statistic (multiple variable)
summarize(Y1+Y2 ~ X1, data = d)
## categorical variable
summarize(treat ~ 1, data = d)
summarize(treat ~ 1, skip.reference = TRUE, data = d)
## aggregate data
summarize( ~ X1 + treat, data = d)
## user defined summary statistic
summarize(Y1 ~ 1, data = d, FUN = quantile)
summarize(Y1 ~ 1, data = d, FUN = quantile, p = c(0.25,0.75))
## complete case summary statistic
summarize(Y1+Y2 ~ X1, data = d2, na.rm = TRUE)
## shortcut to consider all outcomes with common naming
summarize(. ~ treat, data = d2, na.rm = TRUE, filter = "Y")

#### summarize (long format) ####
dL <- reshape(d2, idvar = "id", direction = "long",
             v.names = "Y", varying = c("Y1","Y2","Y3"))
summarize(Y ~ time + X1, data = dL, na.rm  = TRUE)

## user defined summary statistic (outlier)
summarize(Y ~ time + X1, data = dL, FUN = function(x){
   c(outlier.down = sum(x<mean(x,na.rm=TRUE)-2*sd(x,na.rm=TRUE), na.rm=TRUE),
     outlier.up = sum(x>mean(x,na.rm=TRUE)+2*sd(x,na.rm=TRUE), na.rm=TRUE))
}, na.rm = TRUE)

## user defined summary statistic (auc)
myAUC <- function(Y,time){approxAUC(x = time, y = Y, from = 1, to = 3)}
myAUC(Y = dL[dL$id==1,"Y"], time = dL[dL$id==1,"time"])
summarize(Y ~ id, data = dL, FUN = myAUC, na.rm = TRUE)

## add correlation (see correlate function)
e.S <- summarize(Y ~ time + X1, data = dL, repetition = ~time|id,
                 na.rm = TRUE, columns = add("correlation"), na.rm = TRUE)
e.S

#### summarize (long format, missing lines) ####
## use repetition argument to count missing lines in the number of missing values
dL.NNA <- dL[rowSums(is.na(dL))==0,]
summarize(Y ~ time + X1, data = dL.NNA, repetition =~time|id, na.rm  = TRUE)
```

summarizeNA *Summarize missing data patterns*

Description

Summarize missing data patterns.

Usage

```
summarizeNA(
  data,
  formula,
  repetition = NULL,
  sep = "",
  newnames = c("variable", "frequency", "missing.pattern", "n.missing"),
  filter = NULL,
  keep.data = TRUE
)
```

Arguments

`data` [data.frame] dataset containing the observations.

`formula` [formula] On the left hand side the variable(s) for which the missing data patterns should be evaluated and on the right hand side the grouping variables. E.g. Y1 ~ Gender will compute missing data pattern w.r.t Y1 for each gender.

`repetition` [formula] Specify the structure of the data when in the long format: the time/repetition variable and the grouping variable, e.g. ~ time|id. When specified the missing data pattern is specific to each variable not present in the formula.

`sep` [character] character used to separate the missing data indicator (0/1) when naming the missing data patterns.

`newnames` [character vector of length 4] additional column containing the variable name (only when argument `repetition` is used), variables w.r.t. which missing data patterns are identified, frequency of the missing data pattern in the dataset, name of the missing data pattern in the dataset, and number of missing data per pattern.

`filter` [character] a regular expression passed to `grep` to filter the columns of the dataset. Relevant when using `.` to indicate all other variables.

`keep.data` [logical] should the indicator of missing data per variable in the original dataset per pattern be output.

Value

a data frame

See Also

`autoplot.summarizeNA` for a graphical display.

Examples

```
#### display missing data pattern (wide format) ####
data(gastricbypassW, package = "LMMstar")
e.SNA <- summarizeNA(gastricbypassW)
e.SNA
plot(e.SNA)

## only focus on some variables
eG.SNA <- summarizeNA(gastricbypassW, filter = "glucagon")
eG.SNA
plot(eG.SNA)
summarizeNA(weight3+glucagonAUC3 ~ 1, data = gastricbypassW)

#### display missing data pattern (long format) ####
## example 1 (single group)
data(gastricbypassL, package = "LMMstar")
e.SNAL <- summarizeNA(gastricbypassL, repetition = ~time|id)
e.SNAL
plot(e.SNAL, variable = "glucagonAUC")

## example 2 (two groups)
data(calciumL, package = "LMMstar")

## over both groups
mp <- summarizeNA(calciumL, repetition = ~visit|girl)
plot(mp, variable = "bmd")
plot(mp, variable = "bmd", order.pattern = "frequency")
plot(mp, variable = "bmd", order.pattern = 5:1)

## per group
mp2 <- summarizeNA(bmd ~ grp, data = calciumL, repetition = ~visit|girl)
mp2
plot(mp2)

## artificially create different patterns in each group
calciumL2 <- calciumL[order(calciumL$girl),]
calciumL2[calciumL2$girl == 101,"bmd"] <- c(NA,NA,1,1,1)
calciumL2[calciumL2$girl == 104,"bmd"] <- c(NA,1,NA,1,NA)
mp3 <- summarizeNA(bmd ~ grp, data = calciumL2, repetition = ~visit|girl)
mp3
plot(mp3)
plot(mp3, order.pattern = "n.missing")
plot(mp3, order.pattern = "frequency")
```

summary.lmm *Summary Output for a Linear Mixed Model*

Description

Summary output for a linear mixed model fitted with `lmm`.

Usage

```
## S3 method for class 'lmm'
summary(
  object,
  level = 0.95,
  robust = FALSE,
  df = NULL,
  print = TRUE,
  columns = NULL,
  digits = 3,
  digits.df = 1,
  digits.p.value = 3,
  hide.data = FALSE,
  hide.fit = FALSE,
  hide.cor = NULL,
  type.cor = NULL,
  hide.var = NULL,
  hide.sd = NULL,
  hide.re = NULL,
  hide.mean = FALSE,
  ...
)
```

Arguments

`object` [lmm] output of the `lmm` function.

`level` [numeric,0-1] confidence level for the confidence intervals.

`robust` [logical] Should robust standard errors (aka sandwich estimator) be output instead of the model-based standard errors. Can also be `2` compute the degrees-of-freedom w.r.t. robust standard errors instead of w.r.t. model-based standard errors.

`df` [logical] Should a Student's t-distribution be used to model the distribution of the coefficient. Otherwise a normal distribution is used.

`print` [logical] should the output be printed in the console.

`columns` [character vector] Columns to be output for the fixed effects. Can be any of `"estimate"`, `"se"`, `"statistic"`, `"df"`, `"null"`, `"lower"`, `"upper"`, `"p.value"`.

`digits` [interger, >0] number of digits used to display estimates.

`digits.df` [interger, >0] number of digits used to display degrees-of-freedom.

`digits.p.value` [interger, >0] number of digits used to display p-values.

`hide.data` [logical] should information about the dataset not be printed.

`hide.fit` [logical] should information about the model fit not be printed.

`hide.cor` [logical] should information about the correlation structure not be printed.

`type.cor` [character] should the correlation matrix be display (`"matrix"`) or the parameter values (`"param"`).

`hide.var` [logical] should information about the variance not be printed.

`hide.sd` [logical] should information about the standard deviation not be printed.

`hide.re` [logical] should information about the random effect not be printed.

`hide.mean` [logical] should information about the mean structure not be printed.

`...` not used. For compatibility with the generic function.

Value

A list containing elements displayed in the summary:

- `correlation`: the correlation structure.
- `variance`: the variance structure.
- `sd`: the variance structure expressed in term of standard deviations.
- `mean`: the mean structure.

summary.mlmm *Summary of Multiple Linear Mixed Models*

Description

Estimates, p-values, and confidence intevals for multiple linear mixed models.

Usage

```
## S3 method for class 'mlmm'
summary(
  object,
  digits = 3,
  method = NULL,
  print = NULL,
  hide.data = FALSE,
  hide.fit = FALSE,
  ...
)
```

Arguments

`object` an `mlmm` object, output of `mlmm`.

`digits` [integer,>0] number of digits used to display numeric values.

`method` [character] type of adjustment for multiple comparisons, one of `"none"`, `"bonferroni"`, ..., `"fdr"`, `"single-step"`, `"single-step2"`. and/or method(s) to pool the estimates, one of `"average"`, `"pool.se"`, `"pool.gls"`, `"pool.gls1"`, `"pool.rubin"`.

`print` [logical] should the output be printed in the console. Can be a vector of length 2 where the first element refer to the global tests and the second to the individual tests.

`hide.data`	[logical] should information about the dataset not be printed.
`hide.fit`	[logical] should information about the model fit not be printed.
`...`	other arguments are passed to `summary.Wald_lmm`.

`summary.partialCor` *Summary for partial correlation*

Description

Display estimated partial correlation and associated p-values and confidence intevals.

Usage

```
## S3 method for class 'partialCor'
summary(object, digits = 3, detail = TRUE, ...)
```

Arguments

`object`	a `partialCor` object, output of `partialCor`.
`digits`	[integer,>0] number of digits used to display numeric values.
`detail`	[integer,>0] passed to `print.confint_lmm`. If above 0.5 also display when a back-transformation has been used.
`...`	other arguments are passed to `print.confint_lmm`.

`summary.Wald_lmm` *Summary of Testing for a Linear Mixed Models*

Description

Estimates, p-values, and confidence intevals for linear hypothesis testing, possibly adjusted for multiple comparisons.

Usage

```
## S3 method for class 'Wald_lmm'
summary(
  object,
  print = TRUE,
  seed = NULL,
  columns = NULL,
  legend = TRUE,
  digits = 3,
  digits.df = 1,
  digits.p.value = 3,
  sep = ": ",
  ...
)
```

Arguments

`object` an `Wald_lmm` object, output of `anova`.

`print` [logical] should the output be printed in the console. Can be a vector of length 2 where the first element refer to the global tests and the second to the individual tests.

`seed` [integer] value that will be set before adjustment for multiple comparisons to ensure reproducible results. Can also be `NULL`: in such a case no seed is set.

`columns` [character vector] Columns to be displayed for each null hypothesis. Can be any of `"type"`, `"estimate"`, `"se"`, `"statistic"`, `"df"`, `"null"`, `"lower"`, `"upper"`, `"p.value"`.##'

`legend` [logical] should explanations about the content of the table be displayed.

`digits` [interger, >0] number of digits used to display estimates.

`digits.df` [interger, >0] number of digits used to display degrees-of-freedom.

`digits.p.value` [interger, >0] number of digits used to display p-values.

`sep` [character] character string used to separate the type of test (e.g. mean, variance) and the name of the test.

`...` arguments `method`, `level`, and `backtransform` passed to `confint.Wald_lmm`

Details

By default a single step max-test adjustment adjustment is performed in presence of multiple comparisons. It is carried out either using the multcomp package (equal degrees-of-freedom, `method="single-step"`) or using the copula package (unequal degrees-of-freedom, `method="single-step2"`). See the argument `method` of `confint.Wald_lmm` for other adjustments for multiple comparisons.

When considering multiple multivariate Wald tests, adjustment for multiple comparisons for the univariate Wald tests is performed within each multivariate Wald test. The number of tests ajusted for equal the first degree-of-freedom of the multivariate Wald statistic.

Adding the value `"type"` in argument `"columns"` ensures that the type of parameter that is being test (mean, variance, correlation) is output.

Value

`NULL`

`swabsL` *Data From The SWABS Study (Long Format)*

Description

Data from the swabs study, where the pneumococcus was studied in 18 families with different space available for the household. This dataset is in the long format (i.e. one line per measurement).

- `crowding`: space available in the household.
- `family`: family serial number
- `name`: type of family member.
- `swabs`: number of times the swab measurement was positive.

Usage

```
data(swabsL)
```

References

TODO

---

swabsW *Data From The SWABS Study (Wide Format)*

Description

Data from the swabs study, where the pneumococcus was studied in 18 families with different space available for the household. This dataset is in the wide format (i.e. one line per patient).

- `crowding`: space available in the household.
- `family`: family serial number
- `mother`: number of times the swab measurement was positive for the mother.
- `father`: number of times the swab measurement was positive for the father.
- `child1`: number of times the swab measurement was positive for the first child.
- `child2`: number of times the swab measurement was positive for the second child.
- `child3`: number of times the swab measurement was positive for the third child.

Usage

```
data(swabsW)
```

References

Grundy SM, Lan SP, Lachin J. The effects of chenodiol on biliary lipids and their association with gallstone dissolution in the National Cooperative Gallstone Study (SWABS). J Clin Invest. 1984 Apr;73(4):1156-66. doi: 10.1172/JCI111301.

terms.lmm *Model Terms For Linear Mixed Models*

Description

Model terms for linear mixed models. Used by `multcomp::glht`.

Usage

```
## S3 method for class 'lmm'
terms(x, ...)
```

Arguments

`x` a `lmm` object

`...` not used, for compatibility with the generic method.

Value

An object of class `terms` giving a symbolic representation of the mean structure.

TOEPLITZ *Toeplitz Structure*

Description

Variance-covariance structure where the correlation depends on time elapsed between two repetitions. Can be stratified on a categorical variable.

Usage

```
TOEPLITZ(formula, var.cluster, var.time, type = "LAG", add.time)
```

Arguments

`formula` formula indicating on which variable to stratify the residual variance and correlation (left hand side) and variables influencing the residual variance and correlation (right hand side).

`var.cluster` [character] cluster variable.

`var.time` [character] time variable.

`type` [character] degree of flexibility of the correlation structure within covariate (`"UN"`,`"LAG"`,`"CS"`). Will also affect the variance structure when not explicit.

`add.time` Should the default formula (i.e. when `NULL`) contain a time effect.

Details

formula: there can only be at most one covariate for the correlation structure. A typical formula would be ~1, indicating a variance constant over time and a correlation specific to each gap time.

type: for a binary covariate the correlation matrix can be decomposed into four blocs: A, B, B, C. A correspond the correlation within level 0 of the covariate, C within level 1, and B between level 0 and 1. Different correlation structures can be specified:

- "UN": unstructured matrix except for the diagonal elements of C which are constrained to be equal.
- "LAG": Toeplitz structure within A, B, and C, i.e. correlation specific to each time lag and covariate level.
- "CS": block-specific value except for C which has a different value for its diagonal elements.

Value

An object of class TOEPLITZ that can be passed to the argument structure of the lmm function.

Examples

```
## no covariate
TOEPLITZ(~time, var.cluster = "id", var.time = "time")
TOEPLITZ(gender~time, var.cluster = "id", var.time = "time")
TOEPLITZ(list(~time,~time), var.cluster = "id", var.time = "time")
TOEPLITZ(list(gender~time,gender~time), var.cluster = "id", var.time = "time")

## with covariates
TOEPLITZ(~side, var.cluster = "id", type = "UN",
         var.time = "time", add.time = TRUE)
TOEPLITZ(~side, var.cluster = "id", type = "LAG",
         var.time = "time", add.time = TRUE)
TOEPLITZ(~side, var.cluster = "id", type = "CS",
         var.time = "time", add.time = TRUE)
TOEPLITZ(gender~side, var.cluster = "id", type = "CS",
         var.time = "time", add.time = TRUE)
```

UN *Unstructured Structure*

Description

Variance-covariance structure where the residuals have time-specific variance and correlation. Can be stratified on a categorical variable.

Usage

```
UN(formula, var.cluster, var.time, add.time)
```

Arguments

`formula` formula indicating on which variable to stratify the covariance structure.

`var.cluster` [character] cluster variable.

`var.time` [character] time variable.

`add.time` Should the default formula (i.e. when `NULL`) contain a time effect.

Details

A typical formula would be `~1`, indicating a time-specific variance parameter and a correlation parameter specific to each pair of times.

Value

An object of class `UN` that can be passed to the argument `structure` of the `lmm` function.

Examples

```
UN(NULL, var.cluster = "id", var.time = "time", add.time = TRUE)
UN(~gender, var.cluster = "id", var.time = "time", add.time = TRUE)
UN(gender ~ 1, var.cluster = "id", var.time = "time", add.time = TRUE)
UN(list(~gender,~1), var.cluster = "id", var.time = "time", add.time = TRUE)
UN(list(gender~age,gender~1), var.cluster = "id", var.time = "time", add.time = TRUE)
```

`variable.names.lmm` *Variables Involved in a Linear Mixed Model*

Description

Extract the variables used by the linear mixed model.

Usage

```
## S3 method for class 'lmm'
variable.names(object, effects = "all", original = TRUE, simplify = TRUE, ...)
```

Arguments

`object` a `lmm` object.

`effects` [character] Should all variable be output (`"all"`), or only those related to the outcome (`"outcome"`), mean (`"mean"`), variance (`"variance"`), correlation (`"correlation"`), time (`"time"`), cluster (`"cluster"`), strata (`"strata"`)?

`original` [logical] Should only the variables present in the original data be output? When `FALSE`, variables internally created are output instead of the original variable for time, cluster, and strata.

`simplify` [logical] Should the list be converted into a vector if a single `effects` is requested?

`...` not used. For compatibility with the generic function

Value

A list of character vectors or a character vector.

variable.names.mlmm *Variables Involved in Multiple Linear Mixed Models*

Description

Extract the variables used to obtain multiple linear mixed models.

Usage

```
## S3 method for class 'mlmm'
variable.names(object, effects = "all", original = TRUE, simplify = TRUE, ...)
```

Arguments

`object`	a `mlmm` object.
`effects`	[character] Should all variable be output (`"all"`), or only those related to the outcome (`"outcome"`), mean (`"mean"`), variance (`"variance"`), correlation (`"correlation"`), time (`"time"`), cluster (`"cluster"`), strata (`"strata"`), or variable defining the split of the dataset on which a separate linear mixed model is fit (`"by"`)?
`original`	[logical] Should only the variables present in the original data be output? When `FALSE`, variables internally created are output instead of the original variable for time, cluster, and strata.
`simplify`	[logical] Should the list be converted into a vector if a single `effects` is requested?
`...`	not used. For compatibility with the generic function

Value

A list of character vectors or a character vector.

`vasscoresL` *Data From The VAS Study (Long Format)*

Description

Data from the VAS Study, a randomized controlled clinial trial assessing the healing effect of topical zink sulfate on epidermal wound. The study includes 30 heatlhy volunteers with induced wounds on each buttock which where subsequently treated with a different treatment for each wound. Then the VAS-score (pain sensation on a 0-100mm visual analogue scale) was assessed after each treatment application and summarized by area under the curve. This dataset is in the long format (i.e. one line per measurement).

- `id`: patient identifier.
- `group`: treatment group to which the patient has been randomized.
- `treat.num`:
- `vas`: VAS-score relative to the wound.
- `treatment`: Treatment used on the wound. A: active treatment (zink shower gel), B: placebo treatment (shower gel without zink), C: control treatment (demineralized water).

Usage

```
data(vasscoresL)
```

References

TODO

`vasscoresW` *Data From The VAS Study (Wide Format)*

Description

Data from the VAS Study, a randomized controlled clinial trial assessing the healing effect of topical zink sulfate on epidermal wound. The study includes 30 heatlhy volunteers with induced wounds on each buttock which where subsequently treated with a different treatment for each wound. Then the VAS-score (pain sensation on a 0-100mm visual analogue scale) was assessed after each treatment application and summarized by area under the curve. This dataset is in the wide format (i.e. one line per patient).

- `id`: patient identifier.
- `group`: treatment group to which the patient has been randomized.
- `vasA`: VAS-score when using a zink shower gel.
- `vasB`: VAS-score when using a placebo treatment (shower gel without zink).
- `vasC`: VAS-score when using a control treatment with demineralized water.

Usage

```
data(vasscoresW)
```

References

TODO

vcov.lmm *Extract The Variance-Covariance Matrix From a Linear Mixed Model*

Description

Extract the variance-covariance matrix of the model coefficients of a linear mixed model.

Usage

```
## S3 method for class 'lmm'
vcov(
  object,
  effects = NULL,
  robust = FALSE,
  df = FALSE,
  strata = NULL,
  newdata = NULL,
  p = NULL,
  type.information = NULL,
  transform.sigma = NULL,
  transform.k = NULL,
  transform.rho = NULL,
  transform.names = TRUE,
  ...
)
```

Arguments

`object` a `lmm` object.

`effects` [character vector] Should the variance-covariance matrix for all coefficients be output (`"all"`), or only for coefficients relative to the mean (`"mean"` or `"fixed"`), or only for coefficients relative to the variance structure (`"variance"`), or only for coefficients relative to the correlation structure (`"correlation"`). Can also contain `"gradient"` to also output the gradient of the Variance-Covariance matrix .

`robust` [logical] Should robust standard errors (aka sandwich estimator) be output instead of the model-based standard errors. Can also be `2` compute the degrees-of-freedom w.r.t. robust standard errors instead of w.r.t. model-based standard errors.

`df` [logical] Should degrees-of-freedom, computed using Satterthwaite approximation, for the model parameters be output.

`strata` [character vector] When not `NULL`, only output the variance-covariance matrix for the estimated parameters relative to specific levels of the variable used to stratify the mean and covariance structure.

`newdata` [data.frame] dataset relative to which the information should be computed. Only relevant if differs from the dataset used to fit the model.

`p` [numeric vector] value of the model coefficients at which to evaluate the variance-covariance matrix. Only relevant if differs from the fitted values.

`type.information`
[character] Should the expected information be used (i.e. minus the expected second derivative) or the observed inforamtion (i.e. minus the second derivative).

`transform.sigma`
[character] Transformation used on the variance coefficient for the reference level. One of `"none"`, `"log"`, `"square"`, `"logsquare"` - see details.

`transform.k` [character] Transformation used on the variance coefficients relative to the other levels. One of `"none"`, `"log"`, `"square"`, `"logsquare"`, `"sd"`, `"logsd"`, `"var"`, `"logvar"` - see details.

`transform.rho` [character] Transformation used on the correlation coefficients. One of `"none"`, `"atanh"`, `"cov"` - see details.

`transform.names`
[logical] Should the name of the coefficients be updated to reflect the transformation that has been used?

`...` Not used. For compatibility with the generic method.

Details

For details about the arguments **transform.sigma**, **transform.k**, **transform.rho**, see the documentation of the coef.lmm function.

Value

A matrix with one column and column per parameter.

- `df=TRUE`: with an attribute `"df"` containing a numeric vector with one element per parameter.
- `effects` includes `"gradient"`: with an attribute `"gradient"` containing a 3 dimensional array with dimension the number of parameters.

vcov.mlmm *Extract The Variance-Covariance Matrix From Multiple Linear Mixed Models*

Description

Extract the variance-covariance matrix of the model coefficients from multiple linear mixed models.

Usage

```
## S3 method for class 'mlmm'
vcov(
  object,
  effects = "contrast",
  p = NULL,
  newdata = NULL,
  ordering = "by",
  robust = object$args$robust,
  type.information = object$object$type.information,
  transform.sigma = NULL,
  transform.k = NULL,
  transform.rho = NULL,
  transform.names = TRUE,
  simplify = TRUE,
  ...
)
```

Arguments

`object` a `mlmm` object.

`effects` [character] By default will output the estimates relative to the hypotheses being tested (`"contrast"`). But can also output all model coefficients (`"all"`), or only coefficients relative to the mean (`"mean"` or `"fixed"`), or only coefficients relative to the variance structure (`"variance"`), or only coefficients relative to the correlation structure (`"correlation"`).

`p` [list of numeric vector] list of model coefficients to be used. Only relevant if differs from the fitted values.

`newdata` [NULL] Not used. For compatibility with the generic method.

`ordering` [character] should the output be ordered by type of parameter (`"parameter"`) or by model (`"by"`). Not relevant when `effects="contrast"`.

`robust` [logical] Should robust standard errors (aka sandwich estimator) be output instead of the model-based standard errors. Can also be `2` compute the degrees-of-freedom w.r.t. robust standard errors instead of w.r.t. model-based standard errors.

type.information
: [character] Should the expected information be used (i.e. minus the expected second derivative) or the observed inforamtion (i.e. minus the second derivative).

transform.sigma
: [character] Transformation used on the variance coefficient for the reference level. One of "none", "log", "square", "logsquare" - see details.

transform.k
: [character] Transformation used on the variance coefficients relative to the other levels. One of "none", "log", "square", "logsquare", "sd", "logsd", "var", "logvar" - see details.

transform.rho
: [character] Transformation used on the correlation coefficients. One of "none", "atanh", "cov" - see details.

transform.names
: [logical] Should the name of the coefficients be updated to reflect the transformation that has been used?

simplify
: [logical] Should the column names contain the level of the by variable? Not relevant when `effects=\"contrast\"`.

...
: Not used. For compatibility with the generic method.

vcov.rbindWald_lmm *Extract the Variance-Covariance From Wald Tests For Linear Mixed Models*

Description

Extract the variance-covariance matrix from Wald tests applied to a linear mixed models.

Usage

```
## S3 method for class 'rbindWald_lmm'
vcov(
  object,
  effects = "Wald",
  method = "none",
  df = FALSE,
  ordering = NULL,
  transform.names = TRUE,
  simplify = TRUE,
  ...
)
```

Arguments

object
: a `rbindWald_lmm` object.

effects — [character] should the linear contrasts involved in the Wald test be output ("Wald"), or the value of the linear mixed model parameters ("all")? Can also contain "gradient" to also output the gradient of the Variance-Covariance matrix.

method — [character vector] type of adjustment for multiple comparisons across the linear contrasts (one of "none", "bonferroni", ..., "single-step2"). Only relevant when effects = "Wald".

df — [logical] Should degrees-of-freedom, computed using Satterthwaite approximation, for the model parameters be output?

ordering — [character] should the output be ordered by name of the linear contrast ("contrast") or by model ("model").

transform.names — [logical] Should the name of the coefficients be updated to reflect the transformation that has been used? Only relevant when effects="all".

simplify — [logical] should the output be a vector or a list with one element specific to each possible ordering (i.e. contrast or model).

... — Not used. For compatibility with the generic method.

Value

A matrix with one column and column per parameter.

vcov.Wald_lmm — *Extract the Variance-Covariance Matrix From Wald Tests For Linear Mixed Model*

Description

Extract the variance-covariance matrix of the linear contrasts involved in the Wald test.

Usage

```
## S3 method for class 'Wald_lmm'
vcov(object, effects = "Wald", df = FALSE, transform.names = TRUE, ...)
```

Arguments

object — a Wald_lmm object.

effects — [character vector] should the variance-covariance of the linear contrasts involved in the Wald test be output ("Wald"), or of the linear mixed model parameters ("all")? Can also contain "gradient" to also output the gradient of the Variance-Covariance matrix.

df — [logical] Should degrees-of-freedom, computed using Satterthwaite approximation, for the model parameters be output?

transform.names — [logical] Should the name of the coefficients be updated to reflect the transformation that has been used?

... — Not used. For compatibility with the generic method.

Value

A matrix with one column and column per parameter.

- `df=TRUE`: with an attribute `"df"` containing a numeric vector with one element per parameter.
- `effects` includes `"gradient"`: with an attribute `"gradient"` containing a 3 dimensional array with dimension the number of parameters.

`vitaminL` *Data From The Vitamin Study (Long Format)*

Description

Data from the vitamin Study, a randomized study where the growth of guinea pigs was monitored before and after intake of vitamin E/placebo. The weight of each guinea pig was recorded at the end of week 1, 3, 4, 5, 6, and 7. Vitamin E/placebo is given at the beginning of week 5. This dataset is in the long format (i.e. one line per measurement).

- `group`: treatment group: vitamin or placebo.
- `animal`: identifier
- `weight1`: weight (in g) of the pig at the end of week 1 (before treatment).
- `weight3`: weight (in g) of the pig at the end of week 3 (before treatment).
- `weight4`: weight (in g) of the pig at the end of week 4 (before treatment).
- `weight5`: weight (in g) of the pig at the end of week 5 (after treatment).
- `weight6`: weight (in g) of the pig at the end of week 6 (after treatment).
- `weight7`: weight (in g) of the pig at the end of week 7 (after treatment).

Usage

```
data(vitaminL)
```

References

Crowder and Hand (1990, p. 27) Analysis of Repeated Measures.

vitaminW *Data From The Vitamin Study (Wide Format)*

Description

Data from the vitamin Study, a randomized study where the growth of guinea pigs was monitored before and after intake of vitamin E/placebo. The weight of each guinea pig was recorded at the end of week 1, 3, 4, 5, 6, and 7. Vitamin E/placebo is given at the beginning of week 5. This dataset is in the wide format (i.e. one line per patient).

- `group`: treatment group: vitamin or placebo.
- `animal`: identifier
- `weight1`: weight (in g) of the pig at the end of week 1 (before treatment).
- `weight3`: weight (in g) of the pig at the end of week 3 (before treatment).
- `weight4`: weight (in g) of the pig at the end of week 4 (before treatment).
- `weight5`: weight (in g) of the pig at the end of week 5 (after treatment).
- `weight6`: weight (in g) of the pig at the end of week 6 (after treatment).
- `weight7`: weight (in g) of the pig at the end of week 7 (after treatment).

Usage

```
data(vitaminW)
```

References

TODO

weights.rbindWald_lmm *Extract Weights Used to Pool Estimates*

Description

Extract weights used to pool estimates.

Usage

```
## S3 method for class 'rbindWald_lmm'
weights(object, method, effects = "Wald", transform.names = TRUE, ...)
```

Arguments

`object` a `Wald_lmm` object, output of `anova.lmm`, or `rbind.lmm`, or `mlmm`.

`method` [character] method for combining the estimates, one of `"average"`, `"pool.se"`, `"pool.gls"`, `"pool.gls1"`, `"pool.rubin"`.

`effects` [character] should the weights relative to the Wald test be output (`"Wald"`), or the one relative to the linear mixed model parameters (`"all"`)?

`transform.names` [logical] should the name of the coefficients be updated to reflect the transformation that has been used?

`...` Not used. For compatibility with the generic method.

Value

a numeric vector whose elements sum to 1.

Examples

```
set.seed(10)
dL <- sampleRem(250, n.times = 3, format = "long")

e.mlmm <- mlmm(Y~X1+X2+X6, repetition = ~visit|id, data = dL,
               by = "X4", effects = "X1=0", structure = "CS")
weights(e.mlmm, method = c("average","pool.se","pool.gls"))
```

Index