Package ‘multilevelTools’

April 13, 2025

Title Multilevel and Mixed Effects Model Diagnostics and Effect Sizes

Version 0.2.0

URL https://joshuawiley.com/multilevelTools/,
https://github.com/JWiley/multilevelTools

BugReports https://github.com/JWiley/multilevelTools/issues

Description Effect sizes, diagnostics and performance metrics for
multilevel and mixed effects models.
Includes marginal and conditional 'R2' estimates for linear mixed effects models
based on Johnson (2014) <doi:10.1111/2041-210X.12225>.

License GPL (>= 3)

Depends R (>= 4.1.0)

Imports lme4, lmerTest, data.table (>= 1.14.6), nlme, extraoperators
(>= 0.2.0), JWileymisc (>= 1.4.2), ggplot2, ggpubr, scales,
lavaan, zoo, brms, testthat (>= 3.1.0)

Suggests covr, knitr, rmarkdown

Encoding UTF-8

Config/testthat/edition 3

VignetteBuilder knitr

RoxygenNote 7.3.2

NeedsCompilation no

Author Joshua F. Wiley [aut, cre] (<https://orcid.org/0000-0002-0271-6702>)

Maintainer Joshua F. Wiley <jwiley.psych@gmail.com>

Repository CRAN

Date/Publication 2025-04-13 05:50:02 UTC

Contents

`acfByID` *Estimate the autocorrelation by unit (ID)*

Description

This function estimates the autocorrelation over time in a time series by a higher level unit, given by ID.

Usage

```
acfByID(
  xvar,
  timevar,
  idvar,
  data,
  lag.max = 10L,
  na.function = c("na.approx", "na.spline", "na.locf"),
  ...
)
```

Arguments

`xvar`	A character string giving the variable name of the variable to calculate autocorrelations on.
`timevar`	A character string giving the variable name of the time variable.
`idvar`	A character string giving the variable name of the ID variable. Can be missing if only one time series provided, in which case one will be created.

`data`	A data.table containing the variables used in the formula. This is a required argument. If a data.frame, it will silently coerce to a data.table. If not a data.table or data.frame, it will attempt to coerce, with a message.
`lag.max`	An integer of the maximum lag to estimate. Must be equal to or greater than the number of observations for all IDs in the dataset.
`na.function`	A character string giving the name of the function to use to address any missing data. Functions come from the **zoo** package, and must be one of: "na.approx", "na.spline", "na.locf".
`...`	Additional arguments passed to `zoo`.

Value

A data.table of the estimated autocorrelations by ID and lag

Examples

```
## example 1
dat <- data.table::data.table(
  x = sin(1:30),
  time = 1:30,
  id = 1)
acfByID("x", "time", "id", data = dat)

## example 2
dat2 <- data.table::data.table(
  x = c(sin(1:30), sin((1:30)/10)),
  time = c(1:30, 1:30),
  id = rep(1:2, each = 30))
dat2$x[4] <- NA

res <- acfByID("x", "time", "id", data = dat2, na.function = "na.approx")

ggplot2::ggplot(res, ggplot2::aes(factor(Lag), AutoCorrelation)) +
  ggplot2::geom_boxplot()

## clean up
rm(dat, dat2, res)
```

`APAStyler.modelTest.merMod`
Format results from a linear mixed model

Description

Format results from a linear mixed model

Usage

```
## S3 method for class 'modelTest.merMod'
APAStyler(
  object,
  format = list(FixedEffects = c("%s%s [%s, %s]"), RandomEffects = c("%s",
    "%s [%s, %s]"), EffectSizes = c("%s/%s, %s")),
  digits = 2,
 pcontrol = list(digits = 3, stars = TRUE, includeP = FALSE, includeSign = FALSE,
    dropLeadingZero = TRUE),
  ...
)
```

Arguments

`object`	A list of one (or more) models estimated from lmer
`format`	A list giving the formatting style to be used for the fixed effecvts, random effects, and effect sizes. For the random effects, must be two options, one for when the random effects do not have confidence intervals and one when the random effects do have confidence intervals.
`digits`	A numeric value indicating the number of digits to print. This is still in early implementation stages and currently does not change all parts of the output (which default to 2 decimals per APA style).
`pcontrol`	A list controlling how p values are formatted.
`...`	Additional arguments passed to `confint`. Notably `nsim` and `boot.type` if the bootstrap method is used.

Value

a data table of character data

Examples

```
library(JWileymisc)
data(sleepstudy, package = "lme4")

m1 <- lme4::lmer(Reaction ~ Days + (1 + Days | Subject),
  data = sleepstudy)
m2 <- lme4::lmer(Reaction ~ Days + I(Days^2) + (1 + Days | Subject),
  data = sleepstudy)

testm1 <- modelTest(m1)
testm2 <- modelTest(m2)

APAStyler(testm1)
APAStyler(list(Linear = testm1, Quadratic = testm2))
APAStyler(testm1,
  format = list(
    FixedEffects = "%s, %s (%s, %s)",
    RandomEffects = c("%s", "%s (%s, %s)"),
    EffectSizes = "%s, %s; %s"),
  pcontrol = list(digits = 3, stars = FALSE,
                  includeP = TRUE, includeSign = TRUE,
                  dropLeadingZero = TRUE))



testm1 <- modelTest(m1, method = "profile")
testm2 <- modelTest(m2, method = "profile")

APAStyler(testm1)
APAStyler(list(Linear = testm1, Quadratic = testm2))
APAStyler(testm1,
  format = list(
    FixedEffects = "%s, %s (%s, %s)",
    RandomEffects = c("%s", "%s (%s, %s)"),
    EffectSizes = "%s, %s; %s"),
  pcontrol = list(digits = 3, stars = FALSE,
                  includeP = TRUE, includeSign = TRUE,
                  dropLeadingZero = TRUE))



rm(m1, m2, testm1, testm2)
```

`evaluateLags` *Create lag variables and evaluate models with different number of lags*

Description

This function creates the desired number of lags and tests consecutive models from a model with no lags (lag 0), lag 0 + lag1, etc. and reports model performance. This helps evaluate how many lags are needed.

Usage

```
evaluateLags(formula, lagvar, nlags = 0L, idvar, data, ...)
```

Arguments

`formula`	A `character` string giving the `lmer()` formula to use as a base. The variable to be tested with lags gets added as fixed effects only to this, currently.
`lagvar`	A `character` string giving the name of the variable to test lags for.
`nlags`	An `integer` (e.g., 0L, 3L) giving the number of lags to test. Defaults to 0L but really should be more. Must be a positive integer.
`idvar`	A `character` string giving the name o the ID variable.
`data`	A `data.table` dataset ideally or at least a `data.frame`.
`...`	Additional arguments passed to `lmer`, used to control model fitting.

Details

Currently only linear mixed effects models are allowed.

Examples

```
## these examples are slow to run
data(aces_daily, package = "JWileymisc")

evaluateLags(
 "NegAff ~ Female + Age + BornAUS + (1 | UserID)",
 "STRESS",
 4L,
 "UserID",
 aces_daily)


 ## not run, more complex example with random slope, fails to converge
evaluateLags(
 "NegAff ~ Female + Age + BornAUS + (1 + STRESS | UserID)",
 "STRESS",
 5L,
 "UserID",
 aces_daily)

## use different control to fit model and now converges
strictControl <- lme4::lmerControl(optCtrl = list(
   algorithm = "NLOPT_LN_NELDERMEAD",
   xtol_abs = 1e-10,
   ftol_abs = 1e-10))
evaluateLags(
 "NegAff ~ Female + Age + BornAUS + (1 + STRESS | UserID)",
 "STRESS",
 5L,
 "UserID",
 aces_daily,
control = strictControl)
```

`iccMixed` *Intraclass Correlation Coefficient (ICC) from Mixed Models*

Description

This function estimates the ICC from mixed effects models estimated using **lme4**.

Usage

```
iccMixed(dv, id, data, family = c("gaussian", "binomial"))
```

Arguments

`dv`	A character string giving the variable name of the dependent variable.
`id`	A character vector of length one or more giving the ID variable(s). Can be more than one.
`data`	A data.table containing the variables used in the formula. This is a required argument. If a data.frame, it will silently coerce to a data.table. If not a data.table or data.frame, it will attempt to coerce, with a message.
`family`	A character vector giving the family to use for the model. Currently only supports "gaussian" or "binomial".

Value

A data table of the ICCs

References

For details, see Campbell, M. K., Mollison, J., and Grimshaw, J. M. (2001) <doi:10.1002/1097-0258(20010215)20:3 "Cluster trials in implementation research: estimation of intracluster correlation coefficients and sample size."

Examples

```
iccMixed("mpg", "cyl", mtcars)
iccMixed("mpg", "cyl", data.table::as.data.table(mtcars))
iccMixed("mpg", "cyl", data.table::as.data.table(mtcars), family = "gaussian")
iccMixed("mpg", c("cyl", "am"), data.table::as.data.table(mtcars))
iccMixed("am", "cyl", data.table::as.data.table(mtcars), family = "binomial")
```

`meanDecompose` *Mean decomposition of a variable by group(s)*

Description

This function decomposes a variable in a long data set by grouping factors, such as by ID. If multiple grouping factors are listed, the decomposition is in order from left to right. Residuals from the lowest level are returned.

Usage

```
meanDecompose(formula, data)
```

Arguments

`formula`	A formula of the variables to be used in the analysis. Should have the form: variable ~ groupingfactors.
`data`	A data table or data frame containing the variables used in the formula. This is a required argument.

Value

A list of data tables with the means or residuals

Examples

```
meanDecompose(mpg ~ vs, data = mtcars)
meanDecompose(mpg ~ vs + cyl, data = mtcars)

## Example plotting the results
tmp <- meanDecompose(Sepal.Length ~ Species, data = iris)
do.call(ggpubr::ggarrange, c(lapply(names(tmp), function(x) {
  plot(JWileymisc::testDistribution(tmp[[x]]$X), plot = FALSE, varlab = x)$Density
}), ncol = 1))

rm(tmp)
```

---

meanDeviations *Function to calculate the mean and deviations from mean*

Description

Tiny helper function to calculate the mean and deviations from the mean, both returned as a list. Works nicely with data.table to calculate a between and within variable.

Usage

```
meanDeviations(x, na.rm = TRUE)
```

Arguments

x	A vector, appropriate for the `mean` function.
na.rm	A logical, whether to remove missing or not. Defaults to `TRUE`.

Value

A list of the mean (first element) and deviations from the mean (second element).

Examples

```
## simple example showing what it does
meanDeviations(1:10)

## example use case, applied to a data.table
library(data.table)
d <- as.data.table(iris)
d[, c("BSepal.Length", "WSepal.Length") := meanDeviations(Sepal.Length),
  by = Species]
str(d)

rm(d)
```

`modelCompare.merMod` *Compare two lmer models*

Description

This function provides fit statistics and effect sizes for model comparisons. The models must be nested.

Usage

```
## S3 method for class 'merMod'
modelCompare(model1, model2, ...)
```

Arguments

`model1`	A model estimated by `lmer`.
`model2`	A model estimated by `lmer`.
`...`	Additional arguments, not currently used but included to match generic.

Value

a data table with the fit indices for each model and comparing models to each other.

References

For estimating the marginal and conditional R-squared values, see: Nakagawa, S. and Schielzeth, H. (2013). A general and simple method for obtaining R2 from generalized linear mixed-effects models. Methods in Ecology and Evolution, 4(2), 133-142. as well as: Johnson, P. C. (2014). Extension of Nakagawa & Schielzeth's R2GLMM to random slopes models. Methods in Ecology and Evolution, 5(9), 944-946.

Examples

```
library(JWileymisc)
data(aces_daily, package = "JWileymisc")
m1 <- lme4::lmer(PosAff ~ STRESS + (1 + STRESS | UserID),
  data = aces_daily)
m2 <- lme4::lmer(PosAff ~ STRESS + (1 | UserID),
  data = aces_daily)
m3 <- lme4::lmer(PosAff ~ STRESS + Female + (1 | UserID),
  data = aces_daily)

modelCompare(m1, m2)
modelCompare(m2, m3)

rm(m1, m2, m3)
```

`modelDiagnostics.lme` *modelDiagnostics method for lme objects*

Description

This function creates a number of diagnostics for `lme` models.

Usage

```
## S3 method for class 'lme'
modelDiagnostics(
  object,
  ev.perc = 0.001,
  robust = FALSE,
  distr = "normal",
  standardized = TRUE,
  ...
)
```

Arguments

`object`	A fitted model object of class lme.
`ev.perc`	A real number between 0 and 1 indicating the proportion of the theoretical distribution beyond which values are considered extreme values (possible outliers). Defaults to .001.
`robust`	Whether to use robust mean and standard deviation estimates for normal distribution
`distr`	A character string given the assumed distribution. Passed on to `testDistribution`. Defaults to "normal".
`standardized`	A logical whether to use standardized pearson residuals. Defaults to `TRUE` generally where possible but may depend on method.
`...`	Additional arguments, passed to `residualDiagnostics`.

Value

A logical (`is.modelDiagnostics`) or a modelDiagnostics object (list) for `as.modelDiagnostics` and `modelDiagnostics`.

Examples

```
library(JWileymisc)
sleep[1,1] <- NA
m <- nlme::lme(extra ~ group, data = sleep,
 random = ~ 1 | ID, na.action = "na.omit")

 md <- modelDiagnostics(m, ev.perc = .1)
 md$extremeValues

 plot(md)

data(aces_daily, package = "JWileymisc")
m <- nlme::lme(PosAff ~ STRESS, data = aces_daily,
  random = ~ 1 + STRESS | UserID, na.action = "na.omit")
md <- modelDiagnostics(m, ev.perc = .001)
md$extremeValues
plot(md)


m <- nlme::lme(extra ~ 1, data = sleep, random = ~ 1 | ID/group,
  na.action = "na.omit")

md <- modelDiagnostics(m, ev.perc = .1)
md$extremeValues
plot(md)
rm(m, md, sleep)
```

modelDiagnostics.merMod

modelDiagnostics method for merMod objects

Description

This function creates a number of diagnostics for merMod models.

Usage

```
## S3 method for class 'merMod'
modelDiagnostics(
  object,
  ev.perc = 0.001,
  robust = FALSE,
  distr = "normal",
  standardized = TRUE,
  ...
)
```

Arguments

object	A fitted model object, either of class merMod from the lme4 package or merModLmerTest from the lmerTest package.
ev.perc	A real number between 0 and 1 indicating the proportion of the theoretical distribution beyond which values are considered extreme values (possible outliers). Defaults to .001.

`robust`	Whether to use robust mean and standard deviation estimates for normal distribution
`distr`	A character string given the assumed distribution. Passed on to `testDistribution`. Defaults to "normal".
`standardized`	A logical whether to use standardized residuals. Defaults to `TRUE` generally where possible but may depend on method.
`...`	Additional arguments, passed to `residualDiagnostics`.

Value

A logical (`is.modelDiagnostics`) or a modelDiagnostics object (list) for `as.modelDiagnostics` and `modelDiagnostics`.

Examples

```
library(JWileymisc)
sleep[1,1] <- NA
m <- lme4::lmer(extra ~ group + (1 | ID), data = sleep)

md <- modelDiagnostics(m, ev.perc = .1)
md$extremeValues
class(md)

plot(md)

data(aces_daily, package = "JWileymisc")
m <- lme4::lmer(PosAff ~ STRESS + (1 + STRESS | UserID), data = aces_daily)
md <- modelDiagnostics(m, ev.perc = .1)

#  gm1 <- lme4::glmer(cbind(incidence, size - incidence) ~ period + (1 | herd),
#    data = lme4::cbpp, family = binomial)
# modelDiagnostics(gm1) ## should be an error

rm(m, md, sleep)
```

`modelPerformance.merMod`
modelPerformance method for merMod objects

Description

For pseudo R2 by cluster, the squared correlation between observed and predicted values for each cluster unit is returned. For the overall model, the marginal and conditional R2 are calculated as described in the references.

Usage

```
## S3 method for class 'merMod'
modelPerformance(object, ...)
```

Arguments

`object`	A model from `lmer`.
`...`	Additional arguments, not currently used.

Value

a named vector with the marginal and conditional R2 values, if `CLUSTER = FALSE`, otherwise, a data table with the pseudo R2 for each cluster unit. A list with a `data.table` with the following elements:

Model A character string indicating the model type, here merMod

Estimator A character string indicating whether the model was estimated with REML or ML

N_Obs The number of observations

N_Groups A character string indicating the number of unique units in each grouping/clustering variable.

AIC Akaike Information Criterion

BIC Bayesian Information Criterion

LL log likelihood

LLDF log likelihood degrees of freedom

Sigma Residual standard deviation

MarginalR2 in sample variance explained by the fixed effects

ConditionalR2 in sample variance explained by the fixed and random effects

MarginalF2 Cohen's F2 effect size R2 / (1 - R2) based off the Marginal R2

ConditionalF2 Cohen's F2 effect size R2 / (1 - R2) based off the Conditional R2

References

For estimating the marginal and conditional R-squared values, see: Nakagawa, S. and Schielzeth, H. (2013) <doi:10.1111/j.2041-210x.2012.00261.x> "A general and simple method for obtaining R2 from generalized linear mixed-effects models" and also: Johnson, P. C. (2014) <doi:10.1111/2041-210X.12225> "Extension of Nakagawa & Schielzeth's R2GLMM to random slopes models"

Examples

```
library(JWileymisc)
data(aces_daily, package = "JWileymisc")
m1 <- lme4::lmer(PosAff ~ 1 + (1 | UserID),
  data = aces_daily)
modelPerformance(m1)

m1 <- lme4::lmer(PosAff ~ STRESS + (1 + STRESS | UserID),
  data = aces_daily)
modelPerformance(m1)

rm(m1)
```

modelTest.merMod — *estimate detailed results per variable and effect sizes for both fixed and random effects from lmer models*

Description

This function extends the current `drop1` method for `merMod` class objects from the lme4 package. Where the default method to be able to drop both fixed and random effects at once.

Usage

```
## S3 method for class 'merMod'
modelTest(object, method = c("Wald", "profile", "boot"), control, ...)
```

Arguments

`object`	A `merMod` class object, the fitted result of `lmer`.
`method`	A character vector indicating the types of confidence intervals to calculate. One of “Wald”, “profile”, or “boot”.
`control`	A `lmerControl()` results used to control how models are estimated when updating.
`...`	Additional arguments passed to `confint`

Details

At the moment, the function is aimed to `lmer` models and has very few features for `glmer` or `nlmer` models. The primary motivation was to provide a way to provide an overall test of whether a variable “matters”. In multilevel data, a variable may be included in both the fixed and random effects. To provide an overall test of whether it matters requires jointly testing the fixed and random effects. This also is needed to provide an overall effect size.

The function works by generating a formula with one specific variable or “term” removed at all levels. A model is then fit on this reduced formula and compared to the full model passed in. This is a complex operation for mixed effects models for several reasons. Firstly, `R` has no default mechanism for dropping terms from both the fixed and random portions. Secondly, mixed effects models do not accomodate all types of models. For example, if a model includes only a random slope with no random intercept, if the random slope was dropped, there would be no more random effects, and at that point, `lmer` or `glmer` will not run the model. It is theoretically possible to instead fit the model using `lm` or `glm` but this becomes more complex for certain model comparisons and calculations and is not currently implemented. Marginal and conditional R2 values are calculated for each term, and these are used also to calculate something akin to an f-squared effect size.

This is a new function and it is important to carefully evaluate the results and check that they are accurate and that they are sensible. Check accuracy by viewing the model formulae for each reduced model and checking that those are indeed accurate. In terms of checking whether a result is sensible or not, there is a large literature on the difficulty interpretting main effect tests in the presence of interactions. As it is challenging to detect all interactions, especially ones that are made outside of `R` formulae, all terms are tested. However, it likely does not make sense to report results from dropping a main effect but keeping the interaction term, so present and interpret these with caution.

Examples

```
## these examples are slow to run
library(JWileymisc)
m1 <- lme4::lmer(extra ~ group + (1 | ID),
data = sleep, REML=FALSE)
modelTest(m1)


data(aces_daily, package = "JWileymisc")

strictControl <- lme4::lmerControl(optCtrl = list(
   algorithm = "NLOPT_LN_NELDERMEAD",
   xtol_abs = 1e-10,
   ftol_abs = 1e-10))

m1 <- lme4::lmer(NegAff ~ STRESS + (1 + STRESS | UserID),
  data = aces_daily,
  control = strictControl)
modelTest(m1, method = "profile")

m2 <- lme4::lmer(NegAff ~ STRESS + I(STRESS^2) + (1 + STRESS | UserID),
  data = aces_daily, control = strictControl)

## might normally use more bootstraps but keeping low for faster run
modelTest(m2, method = "boot", nsim = 100)
```

`nEffective` *Estimate the effective sample size from longitudinal data*

Description

This function estimates the (approximate) effective sample size.

Usage

```
nEffective(n, k, icc, dv, id, data, family = c("gaussian", "binomial"))
```

Arguments

`n`	The number of unique/indepedent units of observation
`k`	The (average) number of observations per unit
`icc`	The estimated ICC. If missing, will estimate (and requires that the family argument be correctly specified).
`dv`	A character string giving the variable name of the dependent variable.
`id`	A character vector of length one giving the ID variable.

`data`	A data.table containing the variables used in the formula. This is a required argument. If a data.frame, it will silently coerce to a data.table. If not a data.table or data.frame, it will attempt to coerce, with a message.
`family`	A character vector giving the family to use for the model. Currently only supports “gaussian” or “binomial”.

Value

A data.table including the effective sample size.

References

For details, see Campbell, M. K., Mollison, J., and Grimshaw, J. M. (2001) <doi:10.1002/1097-0258(20010215)20:3 "Cluster trials in implementation research: estimation of intracluster correlation coefficients and sample size."

Examples

```
## example where n, k, and icc are estimated from the data
## provided, partly using iccMixed function
nEffective(dv = "mpg", id = "cyl", data = mtcars)

## example where n, k, and icc are known (or being 'set')
## useful for sensitivity analyses
nEffective(n = 60, k = 10, icc = .6)
```

`omegaSEM` *Calculate multilevel omega reliability*

Description

This function uses multilevel structural equation modelling to calculate between and within reliability using coefficient omega.

Usage

```
omegaSEM(items, id, data, savemodel = FALSE)
```

Arguments

`items`	A character vector giving the variables that map to the items in the scale. Note that these should be reverse scored prior to running this function.
`id`	A character string giving the name of the variable that indicates which rows of the dataset belong to the same person or group for the multilevel analysis.
`data`	A data table or data frame to be used for analysis.
`savemodel`	A logical value indicating whether the underlying model should be saved and returned. Defaults to `FALSE`.

Value

a list with two elements, the first, “Results” contains the estimates for coefficient omega at the within and between level. The next element, “Fit” contains the entire fitted model from lavaan, if `savemodel = TRUE`.

References

Geldhof, G. J., Preacher, K. J., & Zyphur, M. J. (2014) <doi:10.1037/a0032138> "Reliability estimation in a multilevel confirmatory factor analysis framework"

Examples

```
data(aces_daily, package = "JWileymisc")
omegaSEM(
  items = c("COPEPrb", "COPEPrc", "COPEExp"),
  id = "UserID",
  data = aces_daily,
  savemodel = FALSE)
```

`plot.modelDiagnostics.lme`

Plot Diagnostics for an lme model

Description

This function creates a number of diagnostic plots from lme models.

Usage

```
## S3 method for class 'modelDiagnostics.lme'
plot(x, y, plot = TRUE, ask = TRUE, ncol = 1, nrow = 1, ...)
```

Arguments

`x`	A fitted model object from `lme()`.
`y`	Included to match the generic. Not used.
`plot`	A logical value whether or not to plot the results or simply return the graaphical objects.
`ask`	A logical whether to ask before changing plots. Only applies to interactive environments.
`ncol`	The number of columns to use for plots. Defaults to 1.
`nrow`	The number of rows to use for plots. Defaults to 1.
`...`	Included to match the generic. Not used.

Value

a list including plots of the residuals, residuals versus fitted values, and one list for plots of all random effects

Examples

```
library(JWileymisc)
sleep[1,1] <- NA
m <- nlme::lme(extra ~ group, data = sleep, random = ~ 1 | ID, na.action = "na.omit")

md <- modelDiagnostics(m, ev.perc = .1)
md$extremeValues

plot(md)

data(aces_daily, package = "JWileymisc")
m <- nlme::lme(PosAff ~ STRESS, data = aces_daily,
  random = ~ 1 + STRESS | UserID, na.action = "na.omit")

md <- modelDiagnostics(m, ev.perc = .001)
md$extremeValues
plot(md$modelDiagnostics[[2]][[2]])
plot(md, ncol = 2, nrow = 2)
plot(md, ncol = 2, nrow = 3)

rm(m, md, sleep)
```

---

plot.modelDiagnostics.merMod *Plot Diagnostics for an lmer model*

Description

This function creates a number of diagnostic plots from lmer models.

Usage

```
## S3 method for class 'modelDiagnostics.merMod'
plot(x, y, plot = TRUE, ask = TRUE, ncol = 1, nrow = 1, ...)
```

Arguments

x	A fitted model object from `lmer()`.
y	Included to match the generic. Not used.
plot	A logical value whether or not to plot the results or simply return the graaphical objects.
ask	A logical whether to ask before changing plots. Only applies to interactive environments.

`ncol`	The number of columns to use for plots. Defaults to 1.
`nrow`	The number of rows to use for plots. Defaults to 1.
`...`	Included to match the generic. Not used.

Value

a list including plots of the residuals, residuals versus fitted values, and one list for plots of all random effects

Examples

```
library(JWileymisc)
sleep[1,1] <- NA
m <- lme4::lmer(extra ~ group + (1 | ID), data = sleep)

md <- modelDiagnostics(m, ev.perc = .1)
md$extremeValues

data(aces_daily, package = "JWileymisc")
m <- lme4::lmer(PosAff ~ STRESS + (1 + STRESS | UserID), data = aces_daily)

md <- modelDiagnostics(m, ev.perc = .001)
md$extremeValues
plot(md$modelDiagnostics[[2]][[2]])
plot(md, ncol = 2, nrow = 2)
plot(md, ncol = 2, nrow = 3)

rm(m, md, sleep)
```

`R2.merMod` *merMod method for R2*

Description

For pseudo R2 by cluster, the squared correlation between observed and predicted values for each cluster unit is returned. For the overall model, the marginal and conditional R2 are calculated as described in the references.

Usage

```
## S3 method for class 'merMod'
R2(object, cluster = FALSE, ...)
```

Arguments

`object`	A model estimated by `lmer`.
`cluster`	A logical whether to calculate individual pseudo R2 values by cluster unit (if `TRUE`) or the marginal and conditional R2 for the overall model (if `FALSE`, the default).
`...`	Added for consistency with generic. Not currently used.

Value

a named vector with the marginal and conditional R2 values, if `CLUSTER = FALSE`, otherwise, a data table with the pseudo R2 for each cluster unit.

References

For estimating the marginal and conditional R-squared values, see: Nakagawa, S. and Schielzeth, H. (2013) <doi:10.1111/j.2041-210x.2012.00261.x> "A general and simple method for obtaining R2 from generalized linear mixed-effects models" and also: Johnson, P. C. (2014) <doi:10.1111/2041-210X.12225> "Extension of Nakagawa & Schielzeth's R2GLMM to random slopes models"

Examples

```
library(JWileymisc)
data(aces_daily, package = "JWileymisc")
m1 <- lme4::lmer(PosAff ~ STRESS + (1 + STRESS | UserID),
  data = aces_daily)

R2(m1)
R2(m1, cluster = TRUE)

hist(R2(m1, cluster = TRUE)$R2)

rm(m1)
```

`ranefdata` *Create data and plots for brms random effect models*

Description

Create data and plots for brms random effect models

Usage

```
ranefdata(object, usevars, newdata, idvar, CI = 0.95, robust = FALSE)
```

Arguments

`object`	a `brmsfit` objectx
`usevars`	a character vector of random effects to plot
`newdata`	a data.table object with the data used to generate the random effects, this is used as an anchor for the random intercepts so they have a meaningful 0 point
`idvar`	a character string specifying the grouping variable name for the random effects
`CI`	a numeric value between 0 and 1 specifying the interval to use. Defaults to 0.95.
`robust`	a logical value indicating whether to use robust estimates or not. Defaults to FALSE. Passed on to `posterior_summary` and `.summary.ID`.

Value

a list with the following components: * `plot`: a list of ggplot objects * `plotdat`: a list of data.table objects with the data used to generate the plots * `relong`: a data.table object with the random effects in long format * `yhat`: a list of data.table objects with the expected values for the random effects * `usevars`: a character vector of the random effects to plot * `idvar`: a character string specifying the grouping variable name for the random effects

Examples

```
if (FALSE) {
library(data.table)
library(brms)
library(ggpubr)

current.seed <- .Random.seed
set.seed(12345)
nGroups <- 100
nObs <- 20
theta.location <- matrix(rnorm(nGroups * 2), nrow = nGroups, ncol = 2)
theta.location[, 1] <- theta.location[, 1] - mean(theta.location[, 1])
theta.location[, 2] <- theta.location[, 2] - mean(theta.location[, 2])
theta.location[, 1] <- theta.location[, 1] / sd(theta.location[, 1])
theta.location[, 2] <- theta.location[, 2] / sd(theta.location[, 2])
theta.location <- theta.location %*% chol(matrix(c(1.5, -.25, -.25, .5^2), 2))
theta.location[, 1] <- theta.location[, 1] - 2.5
theta.location[, 2] <- theta.location[, 2] + 1

dmixed <- data.table(
  x = rep(rep(0:1, each = nObs / 2), times = nGroups))
  dmixed[, ID := rep(seq_len(nGroups), each = nObs)]

  for (i in seq_len(nGroups)) {
    dmixed[ID == i, y := rnorm(
      n = nObs,
      mean = theta.location[i, 1] + theta.location[i, 2] * x,
      sd = exp(1 + theta.location[i, 1] + theta.location[i, 2] * x))
      ]
  }

## note this model takes several minutes, even on a high performance machine
ls.me <- brm(bf(
  y ~ 1 + x + (1 + x | p | ID),
  sigma ~ 1 + x + (1 + x | p | ID)),
  family = "gaussian",
  data = dmixed, seed = 1234,
  silent = 2, refresh = 0, iter = 2000, warmup = 1000, thin = 1,
  chains = 4L, cores = 4L)

out <- ranefdata(
  ls.me,
  newdata = data.table(ID = dmixed$ID[1], x = 0),
  usevars = c("Intercept", "x", "sigma_Intercept", "sigma_x"),
  idvar = "ID")

do.call(ggarrange, c(out$replots, ncol=2,nrow=2))
do.call(ggarrange, c(out$scatterplots, ncol=2,nrow=3))

## set seed back to what it was
set.seed(current.seed)

## cleanup
rm(current.seed, nGroups, nObs, theta.location, dmixed, ls.me, out)
}
```

---

residualDiagnostics.lme

residualDiagnostics methods for lme objects

Description

residualDiagnostics methods for lme objects

Usage

```
## S3 method for class 'lme'
residualDiagnostics(
  object,
  ev.perc = 0.001,
  robust = FALSE,
  distr = "normal",
  standardized = TRUE,
  cut = 8L,
  quantiles = TRUE,
  ...
)
```

Arguments

`object`	An object with class `lme`.
`ev.perc`	The extreme value percentile to use. Defaults to .001.
`robust`	A logical value, whether to use robust estimates or not. Defaults to `FALSE`.
`distr`	A character string specifying the assumed distribution. Currently “normal”, but future options may be supported in the future.
`standardized`	A logical value whether to use standardized pearson residual values or not. Defaults to `TRUE`.
`cut`	An integer, how many unique predicted values there have to be at least for predicted values to be treated continuously, otherwise they are treated as discrete values. Defaults to 8.

`quantiles`	A logical whether to calculate quantiles for the residuals. Defaults to `TRUE`. If `FALSE`, then do not calculate them. These are based on simple quantiles for each predicted value if the predicted values are few enough to be treated discretely. See `cut` argument. Otherwise they are based on quantile regression. First trying smoothing splines, and falling back to linear quantil regression if the splines fail. You may also want to turn these off if they are not working well, or are not of value in your diagnostics.
`...`	Additional arguments. Not currently used.

Value

A logical (`is.residualDiagnostics`) or a residualDiagnostics object (list) for `as.residualDiagnostics` and `residualDiagnostics`.

Examples

```
library(JWileymisc)
sleep[1,1] <- NA
m <- nlme::lme(extra ~ group, data = sleep, random = ~ 1 | ID,
  na.action = na.omit)

 residualDiagnostics(m)$Residuals

m <- nlme::lme(hp ~ mpg, data = mtcars, random = ~ 1 | cyl,
  na.action = na.omit)
residualDiagnostics(m)$Residuals

rm(m, sleep)
```

`residualDiagnostics.merMod` *residualDiagnostics methods for merMod objects*

Description

residualDiagnostics methods for merMod objects

Usage

```
## S3 method for class 'merMod'
residualDiagnostics(
  object,
  ev.perc = 0.001,
  robust = FALSE,
  distr = "normal",
  standardized = TRUE,
  cut = 8L,
  quantiles = TRUE,
  ...
)
```

Arguments

`object`	An object with class `merMod`. Currently only `lmer()` models are supported.
`ev.perc`	The extreme value percentile to use. Defaults to .001.
`robust`	A logical value, whether to use robust estimates or not. Defaults to `FALSE`.
`distr`	A character string specifying the assumed distribution. Currently "normal", but may expand in the future if `glmer()` models are supported.
`standardized`	A logical value whether to use standardized residual values or not. Defaults to `TRUE`.
`cut`	An integer, how many unique predicted values there have to be at least for predicted values to be treated continuously, otherwise they are treated as discrete values. Defaults to 8.
`quantiles`	A logical whether to calculate quantiles for the residuals. Defaults to `TRUE`. If `FALSE`, then do not calculate them. These are based on simple quantiles for each predicted value if the predicted values are few enough to be treated discretely. See `cut` argument. Otherwise they are based on quantile regression. First trying smoothing splines, and falling back to linear quantil regression if the splines fail. You may also want to turn these off if they are not working well, or are not of value in your diagnostics.
`...`	Additional arguments. Not currently used.

Value

A logical (`is.residualDiagnostics`) or a residualDiagnostics object (list) for `as.residualDiagnostics` and `residualDiagnostics`.

Examples

```
library(JWileymisc)
sleep[1,1] <- NA
m <- lme4::lmer(extra ~ group + (1 | ID), data = sleep)

residualDiagnostics(m)$Residuals

#  gm1 <- lme4::glmer(cbind(incidence, size - incidence) ~ period + (1 | herd),
#    data = lme4::cbpp, family = binomial)
# residualDiagnostics(gm1) ## should be an error

rm(m, sleep)
```

`weighted.sma` *Weighted Simple Moving Average*

Description

This function estimates the simple moving average for a specific window and weights it with a variety of optional decays (e.g., exponential, linear, none). Whether to omit missing data or not is based on the missing threshold, which is a proportion and indicates the tolerance. If the weighted proportion missing exceeds this threshold, then that observvation is missing, otherwise, missing data are excluded and the weighted simple moving average calculated on the non missing data.

Usage

```
weighted.sma(
  x,
  window,
  decay = c("exponential", "linear", "none"),
  alpha,
  missThreshold = 0
)
```

Arguments

`x`	Time series data on which to calculate the weighted simple moving average. It is assumed that these data are in the correct order and that time is equally spaced. Any missing data should be filled in with NAs.
`window`	An integer indicating the size of the window to use. This window will include the current value.
`decay`	A character string indicating the type of decay to use on the weights.
`alpha`	An optional value. Not needed for `decay` = "none", but it is required for the exponential and linear decay. For exponential and linear decay, alpha should range between 0 and 1. 0 will result in no decay.
`missThreshold`	A numeric value indicating the proportion of data that can be missing for a given window before the resulting simple moving average is set to missing. This is a proportion of the weighted data, so not all data points will necessarily be equally weighted.

Value

A numeric vector of the weighted simple moving averages

Examples

```
dweights <- expand.grid(
  time = 0:10,
  alpha = seq(0, 1, by = .1))

library(ggplot2)

ggplot(dweights, aes(time, (1 - alpha)^time, colour = factor(alpha))) +
  geom_line() + geom_point() + theme_bw() +
  scale_x_reverse() +
  theme(legend.position = "bottom") +
  ggtitle("Exponential Decay in Weights")

ggplot(dweights, aes(time, pmax(1 - alpha * time, 0), colour = factor(alpha))) +
  geom_line() + geom_point() + theme_bw() +
  scale_x_reverse() +
  theme(legend.position = "bottom") +
  ggtitle("Linear Decay in Weights")

weighted.sma(c(1, 2, 3, 4, 5),
             window = 3L, decay = "none",
             missThreshold = 0)

weighted.sma(c(1, 2, 3, 4, 5),
             window = 3L, decay = "exponential",
             alpha = 0, missThreshold = 0)

weighted.sma(c(1, 2, 3, 4, 5),
             window = 3L, decay = "linear",
             alpha = 0, missThreshold = 0)

weighted.sma(c(1, 2, 3, 4, 5),
             window = 3L, decay = "exponential",
             alpha = .1, missThreshold = 0)

weighted.sma(c(1, 2, 3, 4, 5),
             window = 3L, decay = "exponential",
             alpha = .5, missThreshold = 0)

weighted.sma(c(1, 2, 3, 4, 5),
             window = 3L, decay = "linear",
             alpha = .1, missThreshold = 0)

weighted.sma(c(1, 2, 3, 4, 5),
             window = 3L, decay = "linear",
             alpha = .3, missThreshold = 0)

weighted.sma(c(1, NA, NA, 4, 5),
             window = 4L, decay = "exponential",
             alpha = .4, missThreshold = .4)

## clean up
rm(dweights)
```

Index