Package ‘merTools’

June 23, 2020

Title Tools for Analyzing Mixed Effect Regression Models

Version 0.5.2

Description Provides methods for extracting results from mixed-effect model
objects fit with the 'lme4' package. Allows construction of prediction intervals
efficiently from large scale linear and generalized linear mixed-effects models.

Depends R (>= 3.0.2), arm, lme4 (>= 1.1-11), methods

Suggests testthat, knitr, rmarkdown, parallel, nlme, future.apply,
 rstanarm, Amelia, DT

Imports dplyr, mvtnorm, foreach, shiny, abind, ggplot2, blme,
 broom.mixed,

License GPL (>= 2)

LazyData true

VignetteBuilder knitr

RoxygenNote 7.1.0

Encoding UTF-8

BugReports https://www.github.com/jknowles/merTools

NeedsCompilation no

Author Jared E. Knowles [aut, cre],
Carl Frederick [aut],
Alex Whitworth [ctb]

Maintainer Jared E. Knowles <jknowles@gmail.com>

Repository CRAN

Date/Publication 2020-06-23 10:30:12 UTC

R topics documented:

 averageObs .......................................................... 3
collapseFrame ...................................................... 3
draw ................................................................. 4
expectedRank ....................................................... 5
R topics documented:

- famlink
- fastdisp
- FEsim
- fetch.merMod.msgs
- findFormFuns
- fixef.merModList
- hasWeights
- ICC
- lmerModList
- merTools
- modelFixedEff
- modelInfo
- modelRandEffStats
- plotFEsim
- plotREsim
- plot_sim_error_chks
- predictInterval
- print.merModList
- print.summary.merModList
- randomObs
- ranef.merModList
- REcorrExtract
- REextract
- REimpact
- REMargins
- REquantile
- REsdExtract
- REsim
- RMSE.merMod
- sanitizeNames
- setup_parallel
- shinyMer
- shuffle
- stripAttributes
- subBoot
- subsetList
- sum.mm
- summary.merModList
- superFactor
- thetaExtract
- VarCorr.merModList
- wiggle

Index 42
averageObs

Find the average observation for a merMod object

Description

Extract a data frame of a single row that represents the average observation in a merMod object. This function also allows the user to pass a series of conditioning argument to calculate the average observation conditional on other characteristics.

Usage

averageObs(merMod, varList = NULL, origData = NULL, ...)

Arguments

- **merMod**: a merMod object
- **varList**: optional, a named list of conditions to subset the data on
- **origData**: (default=NULL) a data frame containing the original, untransformed data used to call the model. This MUST be specified if the original variables used in formula function calls are NOT present as 'main effects'.
- **...**: not used currently

Details

Each character and factor variable in the data.frame is assigned to the modal category and each numeric variable is collapsed to the mean. Currently if mode is a tie, returns a "." Uses the collapseFrame function.

Value

A data frame with a single row for the average observation, but with full factor levels. See details for more.

collapseFrame

Collapse a dataframe to a single average row

Description

Take an entire dataframe and summarize it in one row by using the mean and mode.

Usage

collapseFrame(data)
Arguments

data a data.frame

Details

Each character and factor variable in the data.frame is assigned to the modal category and each numeric variable is collapsed to the mean. Currently if mode is a tie, returns a ".".

Value

a data frame with a single row

draw

Draw a single observation out of an object matching some criteria

Description

Draw is used to select a single observation out of an R object. Additional parameters allow the user to control how that observation is chosen in order to manipulate that observation later. This is a generic function with methods for a number of objects.

Usage

draw(object, type = c("random", "average"), varList = NULL, seed = NULL, ...)

# S3 method for class 'merMod'
draw(object, type = c("random", "average"), varList = NULL, seed = NULL, ...)

Arguments

object the object to draw from

string what kind of draw to make. Options include random or average

varList a list specifying filters to subset the data by when making the draw

seed numeric, optional argument to set seed for simulations, ignored if type="average"

... additional arguments required by certain methods

Details

In cases of tie, ".", may be substituted for factors.

Value

a data.frame with a single row representing the desired observation
expectedRank

Examples

```r
fm1 <- lmer(Reaction ~ Days + (Days | Subject), sleepstudy)
# Random case
draw(fm1, type = "random")
# Average
draw(fm1, type = "average")
# Subset
draw(fm1, type = "average", varList = list("Subject" = "308"))
```

expectedRank          Calculate the expected rank of random coefficients that account for uncertainty.

Description

expectedRank calculates the expected rank and the percentile expected rank of any random term in a merMod object. A simple ranking of the estimated random effects (as produced by `ranef`) is not satisfactory because it ignores any amount of uncertainty.

Usage

```r
expectedRank(merMod, groupFctr = NULL, term = NULL)
```

Arguments

- `merMod`: An object of class merMod
- `groupFctr`: An optional character vector specifying the name(s) the grouping factor(s) over which the random coefficient of interest varies. This is the variable to the right of the pipe, `|`, in the [g]lmer formula. This parameter is optional. If none is specified all terms will be returned.
- `term`: An optional character vector specifying the name(s) of the random coefficient of interest. This is the variable to the left of the pipe, `|`, in the [g]lmer formula. Partial matching is attempted on the intercept term so the following character strings will all return rankings based on the intercept (provided that they do not match the name of another random coefficient for that factor): `c("(Intercept)","Int","intercep",...)`.

Details

Inspired by Lingsma et al. (2010, see also Laird and Louis 1989), expectedRank sums the probability that each level of the grouping factor is greater than every other level of the grouping factor, similar to a two-sample t-test.

The formula for the expected rank is:

\[
\text{Expected Rank}_i = 1 + \sum \phi((\theta_i - \theta_k)/\sqrt{\text{var}(\theta_i) + \text{var}(\theta_k))}
\]
where $\phi$ is the standard normal distribution function, $\theta$ is the estimated random effect and $\text{var}(\theta)$ is the posterior variance of the estimated random effect. We add one to the sum so that the minimum rank is one instead of zero so that in the case where there is no overlap between the variances of the random effects (or if the variances are zero), the expected rank equals the actual rank. The ranks are ordered such that the winners have ranks that are greater than the losers.

The formula for the percentile expected rank is:

$$100 \times \left(\frac{\text{ExpectedRank}_i - 0.5}{N_{grp}}\right)$$

where $N_{grp}$ is the number of grouping factor levels. The percentile expected rank can be interpreted as the fraction of levels that score at or below the given level.

NOTE: expectedRank will only work under conditions that lme4::ranef will work. One current example of when this is not the case is for models when there are multiple terms specified per factor (e.g. uncorrelated random coefficients for the same term, e.g. `lmer(Reaction ~ Days + (1 | Subject) + (0 + Days | Subject), data = sleepstudy)`)

**Value**

A data.frame with the following five columns:

- `groupFctr` a character representing name of the grouping factor
- `groupLevel` a character representing the level of the grouping factor
- `term` a character representing the formula term for the group
- `estimate` effect estimate from `lme4::ranef(,condVar=TRUE)`.
- `std.error` the posterior variance of the estimate random effect (from `lme4::ranef(,condVar=TRUE)`); named "term".var.
- `ER` The expected rank.
- `pctER` The percentile expected rank.

**References**


**Examples**

```r
# For a one-level random intercept model
m1 <- lmer(Reaction ~ Days + (1 | Subject), sleepstudy)
(m1.er <- expectedRank(m1))

# For a one-level random intercept model with multiple random terms
m2 <- lmer(Reaction ~ Days + (Days | Subject), sleepstudy)
# ranked by the random slope on Days
(m2.er1 <- expectedRank(m2, term="Days"))
```
# ranked by the random intercept
(m2.er2 <- expectedRank(m2, term="int"))

# For a two-level model with random intercepts
m3 <- lmer(y ~ service * dept + (1|s) + (1|d), InstEval)
# ranked by the random intercept on 's'
(m3.er1 <- expectedRank(m3, groupFctr="s", term="Intercept"))

definition

find link function family

description

find link function family

usage

famlink(object, resp = object@resp)

arguments

object a merMod object
resp the response vector

value

the link function and family

definition

fastdisp: faster display of model summaries

description

Display model fit summary of x or x like objects, fast

usage

fastdisp(x, ...)

## S3 method for class 'merMod'
fastdisp(x, ...)

## S3 method for class 'merModList'
fastdisp(x, ...)

**Arguments**

- `x` a model object
- `...` additional arguments to pass to `arm::display` including number of digits

**Details**

Faster than the implementation in the arm package because it avoids refitting.

The time saving is only noticeable for large, time-consuming (g)lmer fits.

**Value**

A printed summary of a `x` object

**See Also**

`display`

**Examples**

```r
# Compare the time for displaying this modest model
require(arm)
m1 <- lmer(y ~ lectage + studage + (1|d) + (1|s), data=InstEval)
system.time(display(m1))
system.time(fastdisp(m1))
```

---

**FEsim**

*Simulate fixed effects from merMod*

**FEsim** simulates fixed effects from merMod object posterior distributions.

**Description**

Simulate fixed effects from merMod. FEsim simulates fixed effects from merMod object posterior distributions.

**Usage**

`FEsim(merMod, n.sims = 200, oddsRatio = FALSE, seed = NULL)`

**Arguments**

- `merMod` a merMod object from the lme4 package
- `n.sims` number of simulations to use
- `oddsRatio` logical, should parameters be converted to odds ratios?
- `seed` numeric, optional argument to set seed for simulations
Details

Use the Gelman sim technique to build fixed effect estimates and confidence intervals. Uses the sim function in the arm package.

Value

A data frame with the following columns:

- term: Name of fixed term (intercept/coefficient)
- mean: Mean of the simulations
- median: Median of the simulations
- sd: Standard deviation of the simulations, NA if oddsRatio=TRUE

Examples

```r
require(lme4)
m2 <- lmer(Reaction ~ Days + (Days | Subject), sleepstudy)
fe2 <- FEsim(m2, 25)
head(fe2)
```

`fetch.merMod.msgs` 

Extract all warning msgs from a merMod object

Description

Extract all warning msgs from a merMod object

Usage

`fetch.merMod.msgs(x)`

Arguments

- x: a merMod object
findFormFuns

findFormFuns used by averageObs to calculate proper averages

Description

The purpose is to properly derive data for the average observation in the data by being ‘aware’ of formulas that contain interactions and/or function calls. For example, in the old behavior, if the formula contained a square term specified as $I(x^2)$, we were returning the mean of $x^2$ not the square of mean$(x)$.

Usage

findFormFuns(merMod, origData = NULL)

Arguments

merMod  the merMod object from which to draw the average observation
origData (default=NULL) a data frame containing the original, untransformed data used to call the model. This MUST be specified if the original variables used in formula function calls are NOT present as ‘main effects’.

Value

a data frame with a single row for the average observation, but with full factor levels. See details for more.

fixef.merModList

Extract fixed-effects estimates for a merModList

Description

Extract fixed-effects estimates for a merModList

Usage

## S3 method for class 'merModList'
fixef(object, add.dropped = FALSE, ...)

Arguments

object  any fitted model object from which fixed effects estimates can be extracted.
add.dropped for models with rank-deficient design matrix, reconstitute the full-length parameter vector by adding NA values in appropriate locations?
...  optional additional arguments. Currently none are used in any methods.
Details

Extract the estimates of the fixed-effects parameters from a list of fitted merMod models. Takes the mean of the individual fixef objects for each of the component models in the merModList.

Value

a named, numeric vector of fixed-effects estimates.

Examples

```r
sim_list <- replicate(n = 10,
    expr = sleepstudy[sample(row.names(sleepstudy), 180),],
    simplify=FALSE)

fml <- "Reaction ~ Days + (Days | Subject)"

mod <- lmerModList(fml, data = sim_list)

fixef(mod)
```

hasWeights

Identify if a merMod has weights

Description

Identify if a merMod has weights

Usage

```r
hasWeights(merMod)
```

Arguments

merMod the merMod object to test for weights

Value

TRUE or FALSE for whether the model has weights
**Description**

A subset of data from the 1982 High School and Beyond survey used as examples for HLM software

**Usage**

hsb

**Format**

A data frame with 7,185 observations on the following 8 variables.

- **schid**: a numeric vector, 160 unique values
- **mathach**: a numeric vector for the performance on a standardized math assessment
- **female**: a numeric vector coded 0 for male and 1 for female
- **ses**: a numeric measure of student socio-economic status
- **minority**: a numeric vector coded 0 for white and 1 for non-white students
- **schtype**: a numeric vector coded 0 for public and 1 for private schools
- **meanses**: a numeric, the average SES for each school in the data set
- **size**: a numeric for the number of students in the school

**Details**

The data file used for this presentation is a subsample from the 1982 High School and Beyond Survey and is used extensively in Hierarchical Linear Models by Raudenbush and Bryk. It consists of 7,185 students nested in 160 schools.

**Source**

Data made available by UCLA Institute for Digital Research and Education (IDRE) online: [https://stats.idre.ucla.edu/other/hlm/hlm-mlm/introduction-to-multilevel-modeling-using-hlm](https://stats.idre.ucla.edu/other/hlm/hlm-mlm/introduction-to-multilevel-modeling-using-hlm)

**References**


**Examples**

```r
data(hsb)
head(hsb)
```
**ICC**

*Calculate the intraclass correlation using mixed effect models*

**Description**

Calculate the intraclass correlation using mixed effect models

**Usage**

```r
ICC(outcome, group, data, subset = NULL)
```

**Arguments**

- `outcome`  
  a character representing the variable of the outcome
- `group`  
  a character representing the name of the grouping term
- `data`  
  a data.frame
- `subset`  
  an optional subset

**Value**

a numeric for the intraclass correlation

**Examples**

```r
data(sleepstudy)
ICC(outcome = "Reaction", group = "Subject", data = sleepstudy)
```

---

**lmerModList**

*Apply a multilevel model to a list of data frames*

**Description**

Apply a multilevel model to a list of data frames  
Apply a Bayesian multilevel model to a list of data frames  
Apply a generalized linear multilevel model to a list of data frames  
Apply a Bayesian generalized linear multilevel model to a list of data frames

**Usage**

```r
lmerModList(formula, data, parallel = FALSE, ...)
blmerModList(formula, data, parallel = FALSE, ...)
glmerModList(formula, data, parallel = FALSE, ...)
bglmerModList(formula, data, parallel = FALSE, ...)
```
Arguments

- **formula**: a formula to pass through compatible with merMod
- **data**: a list object with each element being a data.frame
- **parallel**: logical, should the models be run in parallel? Default FALSE. If so, the ‘future_lapply’ function from the ‘future.apply’ package is used. See details.
- ... additional arguments to pass to the estimating function

Details

Parallel computing is provided by the ‘futures’ package, and its extension the ‘future.apply’ package to provide the ‘future_lapply’ function for easy parallel computations on lists. To use this package, simply register a parallel backend using the ‘plan()’ function from ‘futures’ - an example is to use ‘plan(multisession)’

Value

- a list of fitted merMod objects of class merModList
- a merModList
- a merModList
- a merModList

Examples

```r
sim_list <- replicate(n = 10,
    expr = sleepstudy[sample(row.names(sleepstudy), 180),],
    simplify=FALSE)
fm1 <- "Reaction ~ Days + (Days | Subject)"
mod <- lmerModList(fm1, data = sim_list)
summary(mod)
```

Description

The merTools package contains convenience tools for extracting useful information from and exploring the implications of merMod objects created by the lme4 package. These convenience functions are especially useful for merMod objects that take a long time to estimate due to their complexity or because they are estimated on very large samples.

Details

See the vignettes for usage examples
modelFixedEff

merMod extraction/utility functions

- `fastdisp`
- `superFactor`
- `REextract`
- `REsim`
- `FEsim`
- `RMSE.merMod`
- `thetaExtract`
- `REquantile`

merMod exploration functions

- `plotREsim`
- `plotFEsim`
- `draw`
- `wiggle`
- `subBoot`
- `predictInterval`
- `expectedRank`
- `REimpact`
- `shinyMer`

```
modelFixedEff

Extract averaged fixed effect parameters across a list of merMod objects
```

Description

Extract averaged fixed effect parameters across a list of merMod objects

Usage

```
modelFixedEff(modList, ...)
```

Arguments

- `modList` an object of class merModList
- `...` additional arguments to pass to `tidy`

Details

The Rubin correction for combining estimates and standard errors from Rubin (1987) is applied to adjust for the within and between imputation variances.
Value

a data.frame of the averaged fixed effect parameters

Examples

```r
sim_list <- replicate(n = 10,
  expr = sleepstudy[sample(row.names(sleepstudy), 180),],
  simplify=FALSE)
fml <- "Reaction ~ Days + (Days | Subject)"
mod <- lmerModList(fml, data = sim_list)
modelFixedEff(mod)
```

---

**Description**

Extract model information from a merMod

**Usage**

```r
modelInfo(object)
```

**Arguments**

- `object` a merMod object

**Value**

Simple summary information about the object, number of observations, number of grouping terms, AIC, and residual standard deviation

**Examples**

```r
sim_list <- replicate(n = 10,
  expr = sleepstudy[sample(row.names(sleepstudy), 180),],
  simplify=FALSE)
fml <- "Reaction ~ Days + (Days | Subject)"
mod <- lmerModList(fml, data = sim_list)
modelInfo(mod[[1]])
lapply(mod, modelInfo)
```
modelRandEffStats

Description

Extract data.frame of random effect statistics from merMod List

Usage

modelRandEffStats(modList)

Arguments

modList

a list of multilevel models

Value

a data.frame

Examples

sim_list <- replicate(n = 10,
    expr = sleepstudy[sample(row.names(sleepstudy), 180),],
    simplify=FALSE)
fml <- "Reaction ~ Days + (Days | Subject)"
mod <- lmerModList(fml, data = sim_list)
modelRandEffStats(mod)

plotFEsim

Plot the results of a simulation of the fixed effects

Description

Plot the simulated fixed effects on a ggplot2 chart

Usage

plotFEsim(
    data,
    level = 0.95,
    stat = "median",
    sd = TRUE,
    intercept = FALSE,
    sigmaScale = NULL,
    oddsRatio = FALSE
)
plotREsim

Arguments

- **data**: a data.frame generated by `FEsim` with simulations of the fixed effects of a `merMod`
- **level**: the width of the confidence interval
- **stat**: a character value indicating the variable name in `data` of the midpoint of the estimated interval, e.g. "mean" or "median"
- **sd**: logical, indicating whether or not to plot error bars around the estimates (default is `TRUE`). Calculates the width of the error bars based on **level** and the variable named "sd" in `data`
- **intercept**: logical, should the intercept be included, default is `FALSE`
- **sigmaScale**: a numeric value to divide the estimate and the standard deviation by in the case of doing an effect size calculation
- **oddsRatio**: logical, should the parameters be converted to odds ratios before plotting

Value

A ggplot2 plot of the coefficient effects

Examples

```r
fm1 <- lmer(Reaction ~ Days + (Days | Subject), sleepstudy)
(p1 <- plotFEsim(FEsim(fm1)))
```

```r
plotREsim(  
data,  
level = 0.95,  
stat = "median",  
sd = TRUE,  
sigmaScale = NULL,  
oddsRatio = FALSE,  
labs = FALSE,  
facet = TRUE  
)
```

Description

Plot the simulated random effects on a ggplot2 chart. Points that are distinguishable from zero (i.e. the confidence band based on **level** does not cross the red line) are highlighted. Currently, the plots are ordered according to the grouping factor.

Usage

```r
plotREsim(  
data,  
level = 0.95,  
stat = "median",  
sd = TRUE,  
sigmaScale = NULL,  
oddsRatio = FALSE,  
labs = FALSE,  
facet = TRUE  
)
```
Arguments

- **data**: a data.frame generated by `REsim` with simulations of the random effects of a `merMod`
- **level**: the width of the confidence interval
- **stat**: a character value indicating the variable name in data of the midpoint of the estimated interval, e.g. "mean" or "median"
- **sd**: a logical indicating whether or not to plot error bars around the estimates (default is TRUE). Calculates the width of the error bars based on level and the variable named "sd" in data
- **sigmaScale**: a numeric value to divide the estimate and the standard deviation by in the case of doing an effect size calculation
- **oddsRatio**: logical, should the parameters be converted to odds ratios before plotting
- **labs**: logical, include the labels of the groups on the x-axis
- **facet**: accepts either logical (TRUE) or list to specify which random effects to plot. If TRUE, facets by both groupFctr and term. If list selects the panel specified by the named elements of the list

Value

- a ggplot2 plot of the coefficient effects

Examples

```r
fm1 <- lmer(Reaction ~ Days + (Days | Subject), sleepstudy)
(p1 <- plotREsim(REsim(fm1)))
#Plot just the random effects for the Days slope
(p2 <- plotREsim(REsim(fm1), facet = list(groupFctr = "Subject", term = "Days")))
```

Description

Extract all warning msgs from a `merMod` object

Usage

```r
plot_sim_error_chks(
  type = c("FE", "RE"),
  level = 0.95,
  stat = c("mean", "median"),
  sd = TRUE,
  sigmaScale = NULL,
```
oddsRatio = FALSE,
labs = FALSE,
facet = TRUE
)

Arguments

type: check a fixed or random effect
level: the width of the confidence interval
stat: a character value indicating the variable name in data of the midpoint of the estimated interval, e.g. "mean" or "median"
sd: a logical indicating whether or not to plot error bars around the estimates (default is TRUE). Calculates the width of the error bars based on level and the variable named "sd" in data
sigmaScale: a numeric value to divide the estimate and the standard deviation by in the case of doing an effect size calculation
oddsRatio: logical, should the parameters be converted to odds ratios before plotting
labs: logical, include the labels of the groups on the x-axis
facet: Accepts either logical (TRUE) or list to specify which random effects to plot. If TRUE, facets by both groupFctr and term. If list selects the panel specified by the named elements of the list

predictInterval

Description

This function provides a way to capture model uncertainty in predictions from multi-level models fit with lme4. By drawing a sampling distribution for the random and the fixed effects and then estimating the fitted value across that distribution, it is possible to generate a prediction interval for fitted values that includes all variation in the model except for variation in the covariance parameters, theta. This is a much faster alternative than bootstrapping for models fit to medium to large datasets.

Usage

predictInterval(
  merMod,
  newdata,
  which = c("full", "fixed", "random", "all"),
  level = 0.8,
  n.sims = 1000,
  stat = c("median", "mean"),
  type = c("linear.prediction", "probability"),
  include.resid.var = TRUE,
  returnSims = FALSE,
predictInterval

```
seed = NULL,
.parallel = FALSE,
.paropts = NULL,
fix.intercept.variance = FALSE,
ignore.fixed.terms = NULL
```

Arguments

- **merMod** a merMod object from lme4
- **newdata** a data.frame of new data to predict
- **which** a character specifying what to return, by default it returns the full interval, but you can also select to return only the fixed variation or the random component variation. If full is selected the resulting data.frame will be `nrow(newdata) * number of model levels` long.
- **level** the width of the prediction interval
- **n.sims** number of simulation samples to construct
- **stat** take the median or mean of simulated intervals
- **type** type of prediction to develop
- **include.resid.var** logical, include or exclude the residual variance for linear models
- **returnSims** logical, should all n.sims simulations be returned?
- **seed** numeric, optional argument to set seed for simulations
- **.parallel,** logical should parallel computation be used, default is FALSE
- **.paropts,** -NOT USED: Caused issue #54- a list of additional options passed into the foreach function when parallel computation is enabled. This is important if (for example) your code relies on external data or packages: use the .export and .packages arguments to supply them so that all cluster nodes have the correct environment set up for computing.
- **fix.intercept.variance** logical; should the variance of the intercept term be adjusted downwards to roughly correct for its covariance with the random effects, as if all the random effects are intercept effects?
- **ignore.fixed.terms** a numeric or string vector of indexes or names of fixed effects which should be considered as fully known (zero variance). This can result in under-conservative intervals, but for models with random effects nested inside fixed effects, holding the fixed effects constant intervals may give intervals with closer to nominal coverage than the over-conservative intervals without this option, which ignore negative correlation between the outer (fixed) and inner (random) coefficients.

Details

To generate a prediction interval, the function first computes a simulated distribution of all of the parameters in the model. For the random, or grouping, effects, this is done by sampling from a
predictInterval

multivariate normal distribution which is defined by the BLUP estimate provided by ranef and the associated variance-covariance matrix for each observed level of each grouping terms. For each grouping term, an array is build that has as many rows as there are levels of the grouping factor, as many columns as there are predictors at that level (e.g. an intercept and slope), and is stacked as high as there are number of simulations. These arrays are then multiplied by the new data provided to the function to produce a matrix of yhat values. The result is a matrix of the simulated values of the linear predictor for each observation for each simulation. Each grouping term has such a matrix for each observation. These values can be added to get the estimate of the fitted value for the random effect terms, and this can then be added to a matrix of simulated values for the fixed effect level to come up with n.sims number of possible yhat values for each observation.

The distribution of simulated values is cut according to the interval requested by the function. The median or mean value as well as the upper and lower bounds are then returned. These can be presented either on the linear predictor scale or on the response scale using the link function in the merMod.

Value

a data.frame with three columns:

fit The center of the distribution of predicted values as defined by the stat parameter.
lwr The lower prediction interval bound corresponding to the quantile cut defined in level.
upr The upper prediction interval bound corresponding to the quantile cut defined in level.

If returnSims = TRUE, then the individual simulations are attached to this data.frame in the attribute sim.results and are stored as a matrix.

Note

merTools includes the functions subBoot and thetaExtract to allow the user to estimate the variability in theta from a larger model by bootstrapping the model fit on a subset, to allow faster estimation.

Examples

```r
m1 <- lmer(Reaction ~ Days + (1 | Subject), sleepstudy)
regFit <- predict(m1, newdata = sleepstudy[11, ] ) # a single value is returned
intFit <- predictInterval(m1, newdata = sleepstudy[11, ] ) # bounded values
# Can do glmer
d1 <- cbpp
d1$y <- d1$incidence / d1$size
gm2 <- glmer(y ~ period + (1 | herd), family = binomial, data = d1,
             nAGQ = 9, weights = d1$size)
regFit <- predict(gm2, newdata = d1[1:10, ])
# get probabilities
regFit <- predict(gm2, newdata = d1[1:10, ], type = "response")
intFit <- predictInterval(gm2, newdata = d1[1:10, ], type = "probability")
intFit <- predictInterval(gm2, newdata = d1[1:10, ], type = "linear.prediction")
```
### print.merModList

**Summarize a merMod list**

#### Description

Summarize a merMod list

#### Usage

```r
## S3 method for class 'merModList'
print(x, ...)
```

#### Arguments

- `x` a modList of class merModList
- `...` additional arguments

#### Value

A summary object of model information

#### Examples

```r
sim_list <- replicate(n = 10,
  expr = sleepstudy[sample(row.names(sleepstudy), 180),],
  simplify=FALSE)
fml <- "Reaction ~ Days + (Days | Subject)"
mod <- lmerModList(fml, data = sim_list)
summary(mod)
```

### print.summary.merModList

**Print the summary of a merMod list**

#### Description

Print the summary of a merMod list

#### Usage

```r
## S3 method for class 'summary.merModList'
print(x, ...)
```
randomObs

Arguments

- `merMod`: an object of class `merMod`
- `varList`: optional, a named list of conditions to subset the data on
- `seed`: numeric, optional argument to set seed for simulations

Value

A data frame with a single row for a random observation, but with full factor levels. See details for more.
**ranef.merModList**

Extract random-effects estimates for a merModList

**Description**

Extract random-effects estimates for a merModList

**Usage**

```r
## S3 method for class 'merModList'
ranef(object, ...)
```

**Arguments**

- `object` - an object of a class of fitted models with random effects, typically a `merMod` object.
- `...` - some methods for these generic functions require additional arguments.

**Details**

Extract the estimates of the random-effects parameters from a list of fitted `merMod` models. Takes the mean of the individual `ranef` objects for each of the component models in the `merModList`.

**Value**

A named, numeric vector of random-effects estimates.

**Examples**

```r
code
```
REcorrExtract  

*Extract the correlations between the slopes and the intercepts from a model*

**Description**

Extract the correlations between the slopes and the intercepts from a model

**Usage**

```r
REcorrExtract(model)
```

**Arguments**

- `model` an object that inherits from class `merMod`

**Value**

a numeric vector of the correlations among the effects

**Examples**

```r
fm1 <- lmer(Reaction ~ Days + (Days | Subject), sleepstudy)
REcorrExtract(fm1)
```

---

REextract  

*Extracts random effects*

**Description**

Extracts random effect terms from an lme4 model

**Usage**

```r
REextract(merMod)
```

**Arguments**

- `merMod` a `merMod` object from the lme4 package

**Value**

a data frame with the following columns:

- `groupFctr` The name of the grouping factor associated with the random effects
- `groupID` The level of the grouping factor associated with the random effects
- `term` One column per random effect, the name is derived from the `merMod`
- `term_se` One column per random effect, the name is derived from the `merMod`
Examples

m2 <- lmer(Reaction ~ Days + (Days | Subject), sleepstudy)
rfx <- REextract(m2)
#Note the column names
head(rfx)

REimpact

Calculate the weighted mean of fitted values for various levels of random effect terms.

Description

REimpact calculates the average predicted value for each row of a new data frame across the distribution of expectedRank for a merMod object. This allows the user to make meaningful comparisons about the influence of random effect terms on the scale of the response variable, for user-defined inputs, and accounting for the variability in grouping terms.

Usage

REimpact(merMod, newdata, groupFctr = NULL, term = NULL, breaks = 3, ...)

Arguments

merMod An object of class merMod
newdata a data frame of observations to calculate group-level differences for
groupFctr The name of the grouping factor over which the random coefficient of interest varies. This is the variable to the right of the pipe, |, in the [g]lmer formula. This parameter is optional, if not specified, it will perform the calculation for the first effect listed by ranef.
term The name of the random coefficient of interest. This is the variable to the left of the pipe, |, in the [g]lmer formula. Partial matching is attempted on the intercept term so the following character strings will all return rankings based on the intercept (provided that they do not match the name of another random coefficient for that factor): c("(Intercept)","Int","intercep",...).
breaks an integer representing the number of bins to divide the group effects into, the default is 3; alternatively it can specify breaks from 0-100 for how to cut the expected rank distribution
...
additional arguments to pass to predictInterval

Details

The function predicts the response at every level in the random effect term specified by the user. Then, the expected rank of each group level is binned to the number of bins specified by the user. Finally, a weighted mean of the fitted value for all observations in each bin of the expected ranks is calculated using the inverse of the variance as the weight – so that less precise estimates are
downweighted in the calculation of the mean for the bin. Finally, a standard error for the bin mean is calculated. This function uses the formula for variance of a weighted mean recommended by Cochran (1977).

Value
A data.frame with all unique combinations of the number of cases, rows in the newdata element, and number of bins:

- **case** The row number of the observation from newdata.
- **bin** The ranking bin for the expected rank, the higher the bin number, the greater the expected rank of the groups in that bin.
- **AvgFitWght** The weighted mean of the fitted values for case i in bin k
- **AvgFitWghtSE** The standard deviation of the mean of the fitted values for case i in bin k.
- **nobs** The number of group effects contained in that bin.

References

See Also
- `expectedRank`, `predictInterval`

Examples
```r
# For a one-level random intercept model
m1 <- lmer(Reaction ~ Days + (1 | Subject), sleepstudy)
m1.er <- REimpact(m1, newdata = sleepstudy[1, ], breaks = 2)
# For a one-level random intercept model with multiple random terms
m2 <- lmer(Reaction ~ Days + (Days | Subject), sleepstudy)
# ranked by the random slope on Days
m2.er1 <- REimpact(m2, newdata = sleepstudy[1, ],
                   groupFctr = "Subject", term="Days")
# ranked by the random intercept
m2.er2 <- REimpact(m2, newdata = sleepstudy[1, ],
                   groupFctr = "Subject", term="int")

# You can also pass additional arguments to predictInterval through REimpact
g1 <- lmer(y ~ lectage + studage + (1|d) + (1|s), data=InstEval)
zed <- REimpact(g1, newdata = InstEval[9:12, ], groupFctr = "d", n.sims = 50,
                include.resid.var = TRUE)
zed2 <- REimpact(g1, newdata = InstEval[9:12, ], groupFctr = "s", n.sims = 50,
                 include.resid.var = TRUE)
zed3 <- REimpact(g1, newdata = InstEval[9:12, ], groupFctr = "d", breaks = 5,
                 n.sims = 50, include.resid.var = TRUE)
```
REmargins

Calculate the predicted value for each observation across the distribution of the random effect terms.

Description

REmargins calculates the average predicted value for each row of a new data frame across the distribution of `expectedRank` for a merMod object. This allows the user to make meaningful comparisons about the influence of random effect terms on the scale of the response variable, for user-defined inputs, and accounting for the variability in grouping terms.

Usage

```r
REmargins(
  merMod, 
  newdata = NULL, 
  groupFctr = NULL, 
  term = NULL, 
  breaks = 4, 
  .parallel = FALSE, 
  ...
)
```

Arguments

- **merMod**: An object of class merMod
- **newdata**: a data frame of observations to calculate group-level differences for
- **groupFctr**: The name of the grouping factor over which the random coefficient of interest varies. This is the variable to the right of the pipe, `|`, in the [g]lmer formula. This parameter is optional, if not specified, it will perform the calculation for the first effect listed by `ranef`. If the length is > 1 then the combined effect of all listed groups will calculated and marginalized over co-occurences of those groups if desired.
- **term**: The name of the random coefficient of interest. This is the variable to the left of the pipe, `|`, in the [g]lmer formula. Partial matching is attempted on the intercept term so the following character strings will all return rankings based on the intercept (provided that they do not match the name of another random coefficient for that factor): c("(Intercept)","Int","intercep",...).
- **breaks**: an integer representing the number of bins to divide the group effects into, the default is 3.
- **.parallel**, logical should parallel computation be used, default is TRUE
- **...**: additional arguments to pass to `predictInterval`
Details

The function simulates the
The function predicts the response at every level in the random effect term specified by the user. Then, the expected rank of each group level is binned to the number of bins specified by the user. Finally, a weighted mean of the fitted value for all observations in each bin of the expected ranks is calculated using the inverse of the variance as the weight – so that less precise estimates are downweighted in the calculation of the mean for the bin. Finally, a standard error for the bin mean is calculated.

Value

A data.frame with all unique combinations of the number of cases, rows in the newdata element:

| case | The row number of the observation from newdata. Each row in newdata will be repeated for all unique levels of the grouping_var, term, and breaks. |
| grouping_var | The grouping variable the random effect is being marginalized over. |
| term | The term for the grouping variable the random effect is being marginalized over. |
| breaks | The ntile of the effect size for grouping_var and term |
| original_group_level | The original grouping value for this case |
| fit_combined | The predicted value from predictInterval for this case simulated at the Nth ntile of the expected rank distribution of grouping_var and term |
| upr_combined | The upper bound of the predicted value. |
| lwr_combined | The lower bound of the predicted value. |
| fit_XX | For each grouping term in newdata the predicted value is decomposed into its fit components via predictInterval and these are all returned here |
| upr_XX | The upper bound for the effect of each grouping term |
| lwr_XX | The lower bound for the effect of each grouping term |
| fit_fixed | The predicted fit with all the grouping terms set to 0 (average) |
| upr_fixed | The upper bound fit with all the grouping terms set to 0 (average) |
| lwr_fixed | The lower bound fit with all the grouping terms set to 0 (average) |

References


See Also

expectedRank,predictInterval
Examples

fm1 <- lmer(Reaction ~ Days + (Days | Subject), sleepstudy)
mfx <- REMargins(merMod = fm1, newdata = sleepstudy[1:10,])

# You can also pass additional arguments to predictInterval through REimpact
g1 <- lmer(y ~ lectage + studage + (1|d) + (1|s), data=InstEval)
margin df <- REMargins(g1, newdata = InstEval[20:25, ], groupFctr = c("s"),
            breaks = 4)
margin df <- REMargins(g1, newdata = InstEval[20:25, ], groupFctr = c("d"),
            breaks = 3)

---

REquantile

Identify group level associated with RE quantile

Description

For a user specified quantile (or quantiles) of the random effect terms in a merMod object. This
allows the user to easily identify the observation associated with the nth percentile effect.

Usage

REquantile(merMod, quantile, groupFctr, term = "(Intercept)"

Arguments

- **merMod**: a merMod object with one or more random effect levels
- **quantile**: a numeric vector with values between 0 and 100 for quantiles
- **groupFctr**: a character of the name of the random effect grouping factor to extract quantiles from
- **term**: a character of the random effect to extract for the grouping factor specified. Default is the intercept.

Value

a vector of the level of the random effect grouping term that corresponds to each quantile

Examples

fm1 <- lmer(Reaction ~ Days + (Days | Subject), sleepstudy)
REquantile(fm1, quantile = 0.25, groupFctr = "Subject")
REquantile(fm1, quantile = 0.25, groupFctr = "Subject", term = "Days")
**REsdExtract**

Extract the standard deviation of the random effects from a merMod object

**Description**

Extract the standard deviation of the random effects from a merMod object

**Usage**

REsdExtract(model)

**Arguments**

- **model**
  - an object that inherits from class merMod

**Value**

- a numeric vector for standard deviations of the random effects

**Examples**

```r
fm1 <- lmer(Reaction ~ Days + (Days | Subject), sleepstudy)
REsdExtract(fm1)
```

---

**REsim**

Simulate random effects from merMod REsim simulates random effects from merMod object posterior distributions

**Description**

Simulate random effects from merMod REsim simulates random effects from merMod object posterior distributions

**Usage**

REsim(merMod, n.sims = 200, oddsRatio = FALSE, seed = NULL)

**Arguments**

- **merMod**
  - a merMod object from the lme4 package
- **n.sims**
  - number of simulations to use
- **oddsRatio**
  - logical, should parameters be converted to odds ratios?
- **seed**
  - numeric, optional argument to set seed for simulations
Details

Use the Gelman sim technique to build empirical Bayes estimates. Uses the sim function in the arm package.

Value

a data frame with the following columns:

- **groupFctr**: Name of the grouping factor
- **groupID**: Level of the grouping factor
- **term**: Name of random term (intercept/coefficient)
- **mean**: Mean of the simulations
- **median**: Median of the simulations
- **sd**: Standard deviation of the simulations, NA if `oddsRatio=TRUE`

Examples

```r
require(lme4)
m2 <- lmer(Reaction ~ Days + (Days | Subject), sleepstudy)
re2 <- RESim(m2, 25)
head(re2)
```

---

RMSE.merMod

*Estimate the Root Mean Squared Error (RMSE) for a lmerMod*

Description

Extract the Root Mean Squared Error for a lmerMod object.

Usage

```r
RMSE.merMod(merMod, scale = FALSE)
```

Arguments

- **merMod**: a lmerMod object from the lme4 package

- **scale**: logical, should the result be returned on the scale of response variable standard deviations?

Value

a numeric which represents the RMSE

Examples

```r
require(lme4)
m2 <- lmer(Reaction ~ Days + (Days | Subject), sleepstudy)
RMSE.merMod(m2)
```
sanitizeNames  
Clean up variable names in data frames

Description
Strip out transformations from variable names in data frames

Usage
sanitizeNames(data)

Arguments
data  a data.frame

Value
a data frame with variable names cleaned to remove factor() construction

setup_parallel  
Set up parallel environment

Description
Set up parallel environment

Usage
setup_parallel()

Value
Nothing
shinyMer

Launch a shiny app to explore your merMod interactively

Description

shinyMer launches a shiny app that allows you to interactively explore an estimated merMod using functions from merTools.

Usage

shinyMer(merMod, simData = NULL, pos = 1)

Arguments

- merMod: An object of class "merMod".
- simData: A data.frame to make predictions from (optional). If NULL, then the user can only make predictions using the data in the frame slot of the merMod object.
- pos: The position of the environment to export function arguments to. Defaults to 1, the global environment, to allow shiny to run.

Value

A shiny app

shuffle

Randomly reorder a dataframe

Description

Randomly reorder a dataframe by row

Usage

shuffle(data)

Arguments

- data: a data frame

Value

a data frame of the same dimensions with the rows reordered randomly
stripAttributes  

Remove attributes from a data.frame

Description

Strips attributes off of a data frame that come with a merMod model.frame

Usage

stripAttributes(data)

Arguments

data  a data.frame

Value

a data frame with variable names cleaned to remove all attributes except for names, row.names, and class

subBoot  

Bootstrap a subset of an lme4 model

Description

Bootstrap a subset of an lme4 model

Usage

subBoot(merMod, n = NULL, FUN, R = 100, seed = NULL, warn = FALSE)

Arguments

merMod  a valid merMod object
n  the number of rows to sample from the original data in the merMod object, by default will resample the entire model frame
FUN  the function to apply to each bootstrapped model
R  the number of bootstrap replicates, default is 100
seed  numeric, optional argument to set seed for simulations
warn  logical, if TRUE, warnings from lmer will be issued, otherwise they will be suppressed default is FALSE
Details

This function allows users to estimate parameters of a large merMod object using bootstraps on a subset of the data.

Value

a data.frame of parameters extracted from each of the R replications. The original values are appended to the top of the matrix.

Examples

(fm1 <- lmer(Reaction ~ Days + (Days | Subject), sleepstudy))
resultMatrix <- subBoot(fm1, n = 160, FUN = thetaExtract, R = 20)

subsetList

Subset a data.frame using a list of conditions

Description

Split a data.frame by elements in a list

Usage

subsetList(data, list)

Arguments

data a data.frame
list a named list of splitting conditions

Value

a data frame with values that match the conditions in the list
**summary.merModList**

**Description**

Print the results of a merMod list

**Usage**

```r
## S3 method for class 'merModList'
summary(object, ...)
```

**Arguments**

- `object`: a modList of class merModList
- `...`: additional arguments
superFactor

Value

summary content printed to console

Examples

```r
sim_list <- replicate(n = 10,
    expr = sleepstudy[sample(row.names(sleepstudy), 180),],
    simplify=FALSE)
fml <- "Reaction ~ Days + (Days | Subject)"
mod <- lmerModList(fml, data = sim_list)
print(mod)
```

---

superFactor  
Create a factor with unobserved levels

Description

Create a factor variable and include unobserved levels for compatibility with model prediction functions

Usage

```r
superFactor(x, fullLev)
```

Arguments

- `x` a vector to be converted to a factor
- `fullLev` a vector of factor levels to be assigned to `x`

Value

a factor variable with all observed levels of `x` and all levels of `x` in `fullLev`

Examples

```r
regularFactor <- c("A", "B", "C")
regularFactor <- factor(regularFactor)
levels(regularFactor)
# Now make it super
newLevs <- c("D", "E", "F")
regularFactor <- superFactor(regularFactor, fullLev = newLevs)
levels(regularFactor) # now super
```
thetaExtract

Extract theta parameters from a merMod model

Description

A convenience function that returns the theta parameters for a merMod object.

Usage

thetaExtract(merMod)

Arguments

merMod a valid merMod object

Value

a vector of the covariance, theta, parameters from a merMod

See Also

merMod

Examples

(fm1 <- lmer(Reaction ~ Days + (Days | Subject), sleepstudy))
thetaExtract(fm1) #(a numeric vector of the covariance parameters)

VarCorr.merModList

Extract the variances and correlations for random effects from a merMod list

Description

Extract the variances and correlations for random effects from a merMod list

Usage

## S3 method for class 'merModList'
VarCorr(x, sigma = 1, rdig = 3L)

Arguments

x for VarCorr: a fitted model object, usually an object inheriting from class merMod. For as.data.frame, a VarCorr.merMod object returned from VarCorr.
sigma an optional numeric value used as a multiplier for the standard deviations.
rdig the number of digits to round to, integer
Value

a list with two elements "stddev" and "correlation" for the standard deviations and correlations averaged across models in the list

Examples

```r
sim_list <- replicate(n = 10,
  expr = sleepstudy[sample(row.names(sleepstudy), 180),],
  simplify=FALSE)
fml <- "Reaction ~ Days + (Days | Subject)"
mod <- lmerModList(fml, data = sim_list)
VarCorr(mod)
```

Description

Assign an observation to different values

Usage

```r
wiggle(data, varlist, valueslist)
```

Arguments

- `data`: a data frame with one or more observations to be reassigned
- `varlist`: a character vector specifying the name(s) of the variable to adjust
- `valueslist`: a list of vectors with the values to assign to var

Details

If the variable specified is a factor, then wiggle will return it as a character.

Value

a data.frame with each row assigned to the one of the new variable combinations. All variable combinations are returned, eg wiggling two variables with 3 and 4 variables respectively will return a new dataset with $3 \times 4 = 12$ observations.

Examples

```r
data(iris)
wiggle(iris[3,], varlist = "Sepal.Width", valueslist = list(c(1, 2, 3, 5)))
wiggle(iris[3:5,], "Sepal.Width", valueslist = list(c(1, 2, 3, 5)))
wiggle(iris[3,], c("Sepal.Width", "Petal.Length"), list(c(1,2,3,5), c(3,5,6)))
wiggle(iris[3:5,], c("Sepal.Width", "Petal.Length"), list(c(1,2,3,5), c(3,5,6)))
```
Index

+Topic datasets
  hsb, 12

averageObs, 3, 10
bglmerModList (lmerModList), 13
blmerModList (lmerModList), 13
collapseFrame, 3
display, 8
draw, 4, 15
expectedRank, 5, 15, 27–30
famlink, 7
fastdisp, 7, 15
FEsim, 8, 15, 18
fetch.merMod.msgs, 9
findFormFuns, 10
fixef.merModList, 10
glmerModList (lmerModList), 13
hasWeights, 11
hsb, 12
ICC, 13
lmerModList, 13
merMod, 18, 19, 25, 40
merTools, 14
merTools-package (merTools), 14
modelFixedEff, 15
modelInfo, 16
modelRandEffStats, 17
plot_sim_error_chks, 19
plotFESim, 15, 17
plotREsim, 15, 18
predictInterval, 15, 20, 27–30
print.merModList, 23
print.summary.merModList, 23
randomObs, 24
ranef, 5
ranef.merModList, 25
REcorrExtract, 26
REextract, 15, 26
REimpact, 15, 27
REMargins, 29
REquantile, 15, 31
RESdExtract, 32
RESim, 15, 19, 32
RMSE.merMod, 15, 33
sanitizeNames, 34
setup_parallel, 34
shinyMer, 15, 35
shuffle, 35
stripAttributes, 36
subBoot, 15, 36
subList, 37
sum.mm, 38
summary.merModList, 38
superFactor, 15, 39
thetaExtract, 15, 40
tidy, 15
VarCorr.merModList, 40
wiggle, 15, 41