

# Package ‘lme4’

January 2, 2012

**Version** 0.999375-42

**Date** 2011-10-02

**Title** Linear mixed-effects models using S4 classes

**Author** Douglas Bates <bates@stat.wisc.edu>, Martin Maechler  
<maechler@R-project.org> and Ben Bolker <bbolker@gmail.com>

**Maintainer** <lme4-authors@R-forge.wu-wien.ac.at>

**Description** Fit linear and generalized linear mixed-effects models.

**Depends** methods, R(>= 2.11.1), Matrix(>= 0.9996875-1), lattice

**LinkingTo** Matrix, stats

**Imports** graphics, nlme, stats4, stats

**Suggests** mlmRev, MEMSS, coda, MASS, sfsmisc, MatrixModels

**LazyLoad** yes

**LazyData** yes

**License** GPL (>= 2)

**URL** <http://lme4.r-forge.r-project.org/>

**Repository** CRAN

**Date/Publication** 2011-10-04 15:58:08

## R topics documented:

cake . . . . .	2
cbpp . . . . .	3
cbpp_PB . . . . .	4
Dyestuff . . . . .	5
fixef . . . . .	6
getME . . . . .	7

HPDinterval . . . . .	8
lmer . . . . .	9
lmList . . . . .	11
lmList-class . . . . .	12
mcmcsmamp . . . . .	13
mer-class . . . . .	14
merMCMC-class . . . . .	18
Pastes . . . . .	20
Penicillin . . . . .	21
ranef . . . . .	22
refit . . . . .	23
simulate-mer . . . . .	24
sleepstudy . . . . .	27
sleepstudy_PB . . . . .	28
VarCorr . . . . .	29
VarCorr-class . . . . .	30
VerbAgg . . . . .	30

<b>Index</b>	<b>32</b>
--------------	-----------

---

cake	<i>Breakage angle of chocolate cakes</i>
------	--

---

## Description

Data on the breakage angle of chocolate cakes made with three different recipes and baked at six different temperatures. This is a split-plot design with the recipes being whole-units and the different temperatures being applied to sub-units (within replicates). The experimental notes suggest that the replicate numbering represents temporal ordering.

## Usage

```
data(cake)
```

## Format

A data frame with 270 observations on the following 5 variables.

replicate a factor with levels 1 to 15

recipe a factor with levels A, B and C

temperature an ordered factor with levels 175 < 185 < 195 < 205 < 215 < 225

angle a numeric vector giving the angle at which the cake broke.

temp numeric value of the baking temperature (degrees F).

## Details

The replicate factor is nested within the recipe factor, and temperature is nested within replicate.

**Source**

Original data were presented in Cook (1938), and reported in Cochran and Cox (1957, p. 300). Also cited in Lee, Nelder and Pawitan (2006).

**References**

Cook, F. E. (1938) *Chocolate cake, I. Optimum baking temperature*. Master's Thesis, Iowa State College.

Cochran, W. G., and Cox, G. M. (1957) *Experimental designs*, 2nd Ed. New York, John Wiley & Sons.

Lee, Y., Nelder, J. A., and Pawitan, Y. (2006) *Generalized linear models with random effects. Unified analysis via H-likelihood*. Boca Raton, Chapman and Hall/CRC.

**Examples**

```
str(cake)
print(fm1 <- lmer(angle ~ recipe * temperature + (1|recipe:replicate), cake,
  REML = FALSE), corr = FALSE)
print(fm2 <- lmer(angle ~ recipe + temperature + (1|recipe:replicate), cake,
  REML = FALSE), corr = FALSE)
print(fm3 <- lmer(angle ~ recipe + temp + (1|recipe:replicate), cake,
  REML = FALSE), corr = FALSE)
anova(fm3, fm2, fm1)
```

cbpp

*Contagious bovine pleuropneumonia***Description**

Contagious bovine pleuropneumonia (CBPP) is a major disease of cattle in Africa, caused by a mycoplasma. This dataset describes the serological incidence of CBPP in zebu cattle during a follow-up survey implemented in 15 commercial herds located in the Boji district of Ethiopia. The goal of the survey was to study the within-herd spread of CBPP in newly infected herds. Blood samples were quarterly collected from all animals of these herds to determine their CBPP status. These data were used to compute the serological incidence of CBPP (new cases occurring during a given time period). Some data are missing (lost to follow-up).

**Usage**

```
data(cbpp)
```

**Format**

A data frame with 56 observations on the following 4 variables.

herd A factor identifying the herd (1 to 15).

incidence The number of new serological cases for a given herd and time period.

size A numeric vector describing herd size at the beginning of a given time period.

period A factor with levels 1 to 4.

**Details**

Serological status was determined using a competitive enzyme-linked immuno-sorbent assay (cELISA).

**Source**

Lesnoff, M., Laval, G., Bonnet, P., Abdicho, S., Workalemahu, A., Kifle, D., Peyraud, A., Lancelot, R., Thiaucourt, F. (2004) Within-herd spread of contagious bovine pleuropneumonia in Ethiopian highlands. *Preventive Veterinary Medicine* **64**, 27–40.

**Examples**

```
## response as a matrix
(m1 <- glmer(cbind(incidence, size - incidence) ~ period + (1 | herd),
             cbpp, binomial, verbose = TRUE))
## response as a vector of probabilities and usage of argument "weights"
m2 <- glmer(incidence / size ~ period + (1 | herd), weights = size,
            cbpp, binomial, verbose = TRUE)

## Confirm that these are equivalent:
stopifnot(all.equal(coef(m1), coef(m2)),
          all.equal(ranef(m1), ranef(m2)))

## GLMM with individual-level variability (accounting for overdispersion)
cbpp$obs <- 1:nrow(cbpp)
(m3 <- glmer(cbind(incidence, size - incidence) ~ period +
             (1 | herd) + (1|obs),
             family = binomial, data = cbpp))
```

---

cbpp\_PB

*Stored parametric bootstrap samples for [cbpp](#) data*


---

**Description**

Examples of parametric bootstrap distributions computed from models fitted to the [cbpp](#) data set

**Usage**

```
cbpp_PB
```

**Format**

Contains a vector of 500 computed deviances under the (simulated) null hypothesis

**See Also**

[simulate-mer](#) for examples of what to do with the parametric bootstrap results

**Examples**

```

## PB test of significance of main effect of period
gm1 <- glmer(cbind(incidence, size - incidence) ~ period + (1 | herd),
             family = binomial, data = cbpp)
gm0 <- update(gm1, . ~. -period)

## generic parametric bootstrapping function; return a single simulated deviance
## difference between full ('m1') and reduced ('m0') models under the
## null hypothesis that the reduced model is the true model
pboot <- function(m0,m1) {
  s <- simulate(m0)
  L0 <- logLik(refit(m0,s))
  L1 <- logLik(refit(m1,s))
  2*(L1-L0)
}
obsdev <- c(2*(logLik(gm1)-logLik(gm0)))
## Not run:
## parametric bootstrap test of significance of correlation between
## random effects of '(Intercept)' and Days
## Timing approx. 240 secs on a 2.66 GHz Intel Core Duo laptop
set.seed(1001)
cbpp_PB <- replicate(500,pboot(gm0,gm1))

## End(Not run)

```

---

Dyestuff

*Yield of dyestuff by batch*


---

**Description**

The Dyestuff data frame provides the yield of dyestuff (Naphthalene Black 12B) from 5 different preparations from each of 6 different batches of an intermediate product (H-acid). The Dyestuff2 data were generated data in the same structure but with a large residual variance relative to the batch variance.

**Usage**

```
data(Dyestuff)
```

**Format**

Data frames, each with 30 observations on the following 2 variables.

Batch a factor indicating the batch of the intermediate product from which the preparation was created.

Yield the yield of dyestuff from the preparation (grams of standard color).

## Details

The Dyestuff data are described in Davies and Goldsmith (1972) as coming from “an investigation to find out how much the variation from batch to batch in the quality of an intermediate product (H-acid) contributes to the variation in the yield of the dyestuff (Naphthalene Black 12B) made from it. In the experiment six samples of the intermediate, representing different batches of works manufacture, were obtained, and five preparations of the dyestuff were made in the laboratory from each sample. The equivalent yield of each preparation as grams of standard colour was determined by dye-trial.”

The Dyestuff2 data are described in Box and Tiao (1973) as illustrating “the case where between-batches mean square is less than the within-batches mean square. These data had to be constructed for although examples of this sort undoubtedly occur in practice, they seem to be rarely published.”

## Source

O.L. Davies and P.L. Goldsmith (eds), *Statistical Methods in Research and Production*, 4th ed., Oliver and Boyd, (1972), section 6.4

G.E.P. Box and G.C. Tiao, *Bayesian Inference in Statistical Analysis*, Addison-Wesley, (1973), section 5.1.2

## Examples

```
str(Dyestuff)
dotplot(reorder(Batch, Yield) ~ Yield, Dyestuff,
        ylab = "Batch", jitter.y = TRUE, aspect = 0.3,
        type = c("p", "a"))
dotplot(reorder(Batch, Yield) ~ Yield, Dyestuff2,
        ylab = "Batch", jitter.y = TRUE, aspect = 0.3,
        type = c("p", "a"))
(fm1 <- lmer(Yield ~ 1|Batch, Dyestuff))
(fm2 <- lmer(Yield ~ 1|Batch, Dyestuff2))
```

---

fixef

*Extract the fixed-effects estimates*

---

## Description

Extract the estimates of the fixed-effects parameters from a fitted model.

## Usage

```
fixef(object, ...)
fixed.effects(object, ...) # deprecated
```

## Arguments

**object** any fitted model object from which fixed effects estimates can be extracted.  
**...** optional additional arguments. Currently none are used in any methods.

**Value**

a named, numeric vector of fixed-effects estimates.

**Examples**

```
fixef(lmer(Reaction ~ Days + (1|Subject) + (0+Days|Subject), sleepstudy))
```

---

getME	<i>Extract or Get Generalize Components from a Fitted Mixed Effects Model</i>
-------	---

---

**Description**

Extract (or “get”) “components” – in a generalized sense – from a fitted mixed effects model, i.e. (in this version of the package) from an R object of class “mer”.

The goal is to provide “everything a user may want” from a fitted “mer” object *as far* as it is not available by methods, such as [fixef](#), [ranef](#), [vcov](#), etc.

**Usage**

```
getME(object, name)
```

**Arguments**

object	a fitted mixed-effects model of class “mer”, i.e. typically the result of <a href="#">lmer()</a> , <a href="#">glmer()</a> or <a href="#">nlmer()</a> .
name	a character string specifying the name of the “component”. Note this may not be the name of <a href="#">slot</a> of object.

**Value**

Unspecified, as very much depending on the [name](#).

**See Also**

[getCall\(\)](#) (in R  $\geq$  2.14.0; otherwise in **MatrixModels**).

More standard methods for \*mer() objects, such as [ranef](#), [fixef](#), [vcov](#), etc.

**Examples**

```
## shows many methds you should consider *before* getME():
showMethods(class = "mer")

(fm1 <- lmer(Reaction ~ Days + (Days|Subject), sleepstudy))
Z <- getME(fm1, "Z")
stopifnot(is(Z, "CsparseMatrix"),
          c(180,36) == dim(Z),
```

```

all.equal(fixef(fm1), getME(fm1, "beta"),
  check.attr=FALSE, tol = 0))

## All that can be accessed [potentially ..]:
(nmME <- eval(formals(getME)$name))

```

---

HPDinterval

*Highest Posterior Density intervals*


---

### Description

Create Highest Posterior Density (HPD) intervals for the parameters in an MCMC sample.

### Usage

```
HPDinterval(object, prob = 0.95, ...)
```

### Arguments

object	The object containing the MCMC sample - usually of class <code>merMCMC</code> or a numeric matrix.
prob	A numeric scalar in the interval (0,1) giving the target probability content of the intervals. The nominal probability content of the intervals is the multiple of $1/\text{nrow}(\text{obj})$ nearest to prob.
...	Optional additional arguments for methods. None are used at present.

### Details

For each parameter the interval is constructed from the empirical cdf of the sample as the shortest interval for which the difference in the ecdf values of the endpoints is the nominal probability. Assuming that the distribution is not severely multimodal, this is the HPD interval.

### Value

For an `merMCMC` object, a list of matrices with columns "lower" and "upper" and rows corresponding to the parameters. The attribute "Probability" is the nominal probability content of the intervals.

**Description**

Fit a linear mixed model or a generalized linear mixed model or a nonlinear mixed model.

**Usage**

```
lmer(formula, data, family = NULL, REML = TRUE,
      control = list(), start = NULL, verbose = FALSE,
      doFit = TRUE, subset, weights, na.action, offset,
      contrasts = NULL, model = TRUE, x = TRUE, ...)
lmer2(formula, data, family = NULL, REML = TRUE,
       control = list(), start = NULL, verbose = FALSE,
       subset, weights, na.action, offset,
       contrasts = NULL, model = TRUE, x = TRUE, ...)
glmer(formula, data, family = gaussian, start = NULL,
       verbose = FALSE, nAGQ = 1, doFit = TRUE, subset, weights,
       na.action, offset, contrasts = NULL, model = TRUE,
       control = list(), ...)
nlmer(formula, data, start = NULL, verbose = FALSE, nAGQ = 1,
       doFit = TRUE, subset, weights, na.action,
       contrasts = NULL, model = TRUE, control = list(), ...)
```

**Arguments**

formula	a two-sided linear formula object describing the fixed-effects part of the model, with the response on the left of a <code>~</code> operator and the terms, separated by <code>+</code> operators, on the right. The vertical bar character <code> </code> separates an expression for a model matrix and a grouping factor.
data	an optional data frame containing the variables named in <code>formula</code> . By default the variables are taken from the environment from which <code>lmer</code> is called.
family	a GLM family, see <code>glm</code> and <code>family</code> . If <code>family</code> is missing then a linear mixed model is fit; otherwise a generalized linear mixed model is fit.
REML	logical argument to <code>lmer</code> only. Should the estimates be chosen to optimize the REML criterion (as opposed to the log-likelihood)? Defaults to <code>TRUE</code> .
nAGQ	a positive integer - the number of points per axis for evaluating the adaptive Gauss-Hermite approximation to the log-likelihood. This defaults to 1, corresponding to the Laplacian approximation. Values greater than 1 produce greater accuracy in the evaluation of the log-likelihood at the expense of speed.
control	a list of control parameters. See below for details.
start	a named list of starting values for the parameters in the model. If the list is of the same form as the <code>ST</code> slot, it becomes the starting values of the <code>ST</code> slot. If the list contains components named <code>fixef</code> and/or <code>ST</code> , these are used as the starting

values for those slots. (Setting starting values for `fixef` has no effect for a linear mixed model because the fixed-effects parameters do not appear in the profiled deviance.) In `lmer` and `glmer` a numeric `start` argument of the appropriate length is used as the starting value of the parameter vector that determines the ST slot. In `nlmer` a numeric `start` argument is used as the starting values of the `fixef` slot.

<code>doFit</code>	logical scalar. When <code>doFit = FALSE</code> the model is not fit but instead a structure with the model matrices for the random-effects terms is returned, so they can be modified for special model forms. When <code>doFit = TRUE</code> , the default, the model is fit immediately.
<code>subset</code> , <code>weights</code> , <code>na.action</code> , <code>offset</code> , <code>contrasts</code>	further model specification arguments as in <code>lm</code> ; see there for details.
<code>model</code>	logical scalar. If <code>FALSE</code> the model frame in slot <code>frame</code> is truncated to zero rows.
<code>x</code>	logical scalar. If <code>FALSE</code> the model matrix in slot <code>X</code> is truncated to zero rows.
<code>verbose</code>	logical scalar. If <code>TRUE</code> verbose output is generated during the optimization of the parameter estimates.
<code>...</code>	other potential arguments. A <code>method</code> argument was used in earlier versions of the package. Its functionality has been replaced by the <code>REML</code> and <code>nAGQ</code> arguments.

## Details

The `lmer` and `glmer` functions are nearly interchangeable. If `lmer` is called with a non-default family argument the call is replaced by a call to `glmer` with the current arguments. If `glmer` is called with the default family, namely the `gaussian` family with the identity link, then the call is replaced by a call to `lmer` with the current arguments. (They are described as “nearly” interchangeable because the `REML` argument only applies to calls to `lmer` and the `nAGQ` argument only applies to calls to `glmer`.)

Additional standard arguments to model-fitting functions can be passed to `lmer` or `glmer` or `nlmer`:

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

**na.action** a function that indicates what should happen when the data contain NAs. The default action (`na.fail`) prints an error message and terminate if there are any incomplete observations.

**control** a named list of control parameters for the estimation algorithm, specifying only the ones to be changed from their default values. Hence defaults to an empty list.

Possible control options and their default values are:

**msVerbose:** a logical value passed as the `trace` argument to `nlminb` (see documentation on that function). Default is `getOption("verbose")`.

**maxIter:** a positive integer passed as the `maxIter` argument to `nlminb` (see documentation on that function). Default is 300.

**maxFN:** a positive integer specifying the maximum number of evaluations of the deviance function allowed during the optimization. Default is 900.

**model, x** logicals. If TRUE the corresponding components of the fit (the model frame, the model matrices) are returned.

The `lmer2` name exists only for backwards compatibility. Calling this function simply produces an equivalent call to `lmer`.

### Value

An object of class "`mer`", for which many methods are available. See there for details.

### See Also

The `mer` class, `lm`

### Examples

```
## linear mixed models
(fm1 <- lmer(Reaction ~ Days + (Days|Subject), sleepstudy))
(fm2 <- lmer(Reaction ~ Days + (1|Subject) + (0+Days|Subject), sleepstudy))
anova(fm1, fm2)
## generalized linear mixed model
(gm1 <- glmer(cbind(incidence, size - incidence) ~ period + (1 | herd),
             family = binomial, data = cbpp))
## GLMM with individual-level variability (accounting for overdispersion)
cbpp$obs <- 1:nrow(cbpp)
(gm2 <- glmer(cbind(incidence, size - incidence) ~ period +
             (1 | herd) + (1|obs),
             family = binomial, data = cbpp))
anova(gm1, gm2)
## nonlinear mixed models
(nm1 <- nlmer(circumference ~ SSlogis(age, Asym, xmid, scal) ~ Asym|Tree,
             Orange, start = c(Asym = 200, xmid = 725, scal = 350)))
```

---

lmList

*List of lm Objects with a Common Model*

---

### Description

The data argument is split according to the levels of the grouping factor `g` and individual `lm` or `glm` fits are obtained for each data partition, using the model defined in object.

### Usage

```
lmList(formula, data, family, subset, weights,
       na.action, offset, pool, ...)
```

**Arguments**

formula	a linear formula object of the form $y \sim x_1 + \dots + x_n \mid g$ . In the formula object, $y$ represents the response, $x_1, \dots, x_n$ the covariates, and $g$ the grouping factor specifying the partitioning of the data according to which different <code>lm</code> fits should be performed.
data	a data frame in which to interpret the variables named in <code>object</code> .
family	an optional family specification for a generalized linear model.
weights	an optional vector of weights to be used in the fitting process.
subset	an optional vector specifying a subset of observations to be used in the fitting process.
na.action	a function which indicates what should happen when the data contain NAs. The default is set by the <code>na.action</code> setting of <code>options</code> , and is <code>na.fail</code> if that is unset. The “factory-fresh” default is <code>na.omit</code> .
offset	this can be used to specify an <i>a priori</i> known component to be included in the linear predictor during fitting.
pool	an optional logical value that is preserved as an attribute of the returned value. This will be used as the default for <code>pool</code> in calculations of standard deviations or standard errors for summaries.
...	optional arguments to be passed to the model-fitting function.

**Value**

an object of class "`lmList`", which is a list of `lm` objects with as many components as the number of groups defined by the grouping factor.

**See Also**

`lm`

**Examples**

```
(fm1 <- lmList(Reaction ~ Days | Subject, sleepstudy))
```

---

lmList-class

Class "lmList"

---

**Description**

A list of objects of class `lm` with a common model.

**Objects from the Class**

Objects can be created by calls of the form `new("lmList", ...)` or, more commonly, by a call to `lmList`.

**Slots**

**.Data:** Object of class "list", a list of objects of class lm  
**call:** Object of class "call", the function call used to create the lmList object.  
**pool:** Object of class "logical", if TRUE then calculate the pooled standard deviation estimate when displaying the object.

**Extends**

Class "list", from data part. Class "vector", by class "list".

**Methods**

**coef** signature(object = "lmList"): extract the coefficients for the linear models.  
**formula** signature(x = "lmList"): extract the formula used when creating the lmList object.  
**confint** signature(object = "lmList"): confidence intervals for the fitted linear models.  
**show** signature(object = "lmList"): show the object.  
**update** signature(object = "lmList"): update the fit.  
**plot** signature(object = "lmList.confint"): plot the confidence intervals for each parameter.

---

mcmcsamp

*Generate an MCMC sample*


---

**Description**

This generic function generates a sample from the posterior distribution of the parameters of a fitted model using Markov Chain Monte Carlo methods.

**Usage**

```
mcmcsamp(object, n, verbose, ...)
## S4 method for signature 'mer'
mcmcsamp(object, n, verbose, saveb, ...)
```

**Arguments**

object	An object of a suitable class - usually an "mer" object.
n	integer - number of samples to generate. Defaults to 1.
verbose	logical - if TRUE verbose output is printed. Defaults to FALSE.
saveb	logical - if TRUE the values of the random effects are saved as part of the chain. Default is FALSE. Note that saving the random effects can require a considerable amount of memory. Use with caution.
...	Some methods may take additional, optional arguments.

**Value**

An object of class "merMCMC" for which many methods are defined.

**Examples**

```
(fm1 <- lmer(Reaction ~ Days + (1|Subject) + (0+Days|Subject), sleepstudy))
set.seed(101); samp0 <- mcmcscamp(fm1, n = 1000)
str(samp0)
```

---

mer-class

*Mixed Model Representations and \*mer Methods*


---

**Description**

The mer class represents linear or generalized linear or nonlinear mixed-effects models. It incorporates sparse model matrices for the random effects and corresponding sparse Cholesky factors. The summary.mer class represents the summary of these objects.

**Usage**

```
## Methods with "surprising" arguments
## S4 method for signature 'mer'
deviance(object, REML = NULL, ...)
## S4 method for signature 'mer'
expand(x, sparse = TRUE, ...)
## S4 method for signature 'mer'
logLik(object, REML = NULL, ...)
## S4 method for signature 'mer'
print(x, digits, correlation, symbolic.cor, signif.stars, ...)
```

**Arguments**

object	object of class mer.
REML	logical indicating if REML should be used. A value of NULL, the default, or NA indicates that the REML values should be returned if the model was fit by REML, otherwise the ML values.
x	object of class mer to expand.
sparse	logical scalar indicating if the sparse form of the expanded T and S matrices should be returned.
digits	number of digits to use when printing tables of parameter estimates. Defaults to max(3, getOption("digits") - 3).
correlation	logical - should the correlation matrix of the fixed-effects parameter estimates be printed? Defaults to TRUE.
symbolic.cor	logical - should a symbolic form of the correlation matrix be printed instead of the numeric form? Defaults to FALSE.

signif.stars logical - should the 'significance stars' be printed as part of the table of fixed-effects parameter estimates? Defaults to `getOption("show.signif.stars")`.  
 ... potential further arguments passed to methods.

### Objects from the Class

Objects can be created by calls of the form `new("mer", ...)` or, more commonly, via the `lmer`, `glmer` or `nlmer` functions.

### Slots

The class "mer" represents a linear or generalized linear or nonlinear or generalized nonlinear mixed model and contains the slots:

**env:** An environment (class "environment") created for the evaluation of the nonlinear model function. Not used except by `nlmer` models.

**nlmodel:** The nonlinear model function as an object of class "call". Not used except by `nlmer` models.

**frame:** The model frame (class "data.frame").

**call:** The matched call to the function that created the object. (class "call").

**flist:** The list of grouping factors for the random effects.

**X:** Model matrix for the fixed effects. In an `nlmer` fitted model this matrix has  $n * s$  rows where  $n$  is the number of observations and  $s$  is the number of parameters in the nonlinear model.

**Zt:** The transpose of model matrix for the random effects, stored as a compressed column-oriented sparse matrix (class "dgCMatrix").

**pWt:** Numeric prior weights vector. This may be of length zero (0), indicating unit prior weights.

**offset:** Numeric offset vector. This may be of length zero (0), indicating no offset.

**y:** The response vector (class "numeric").

**Gp:** Integer vector of group pointers within the random effects vector. The elements of Gp are 0-based indices of the first element from each random-effects term. Thus the first element is always 0. The last element is the total length of the random effects vector.

**dims:** A named integer vector of dimensions. Some of the dimensions are  $n$ , the number of observations,  $p$ , the number of fixed effects,  $q$ , the total number of random effects,  $s$ , the number of parameters in the nonlinear model function and  $nt$ , the number of random-effects terms in the model.

**ST:** A list of S and T factors in the TSST' Cholesky factorization of the relative variance matrices of the random effects associated with each random-effects term. The unit lower triangular matrix,  $T$ , and the diagonal matrix,  $S$ , for each term are stored as a single matrix with diagonal elements from  $S$  and off-diagonal elements from  $T$ .

**V:** Numeric gradient matrix (class "matrix") of the nonlinear model function. Not used except by `nlmer` models.

**A:** Scaled sparse model matrix (class "dgCMatrix") for the the unit, orthogonal random effects,  $U$ .

**Cm:** Reduced, weighted sparse model matrix (class "dgCMatrix") for the unit, orthogonal random effects,  $U$ . Not used except by `nlmer` models.

- Cx:** The "x" slot in the weighted sparse model matrix (class "[dgCMatrix](#)") for the unit, orthogonal random effects,  $U$ , in generalized linear mixed models. For these models the matrices  $A$  and  $C$  have the same sparsity pattern and only the "x" slot of  $C$  needs to be stored.
- L:** The sparse lower Cholesky factor of  $P(AA' + I)P'$  (class "[dCHMfactor](#)") where  $P$  is the fill-reducing permutation calculated from the pattern of nonzeros in  $A$ .
- deviance:** Named numeric vector containing the deviance corresponding to the maximum likelihood (the "ML" element) and "REML" criteria and various components. The "ldL2" element is twice the logarithm of the determinant of the Cholesky factor in the L slot. The "usqr" component is the value of the random-effects quadratic form.
- fixef:** Numeric vector of fixed effects.
- ranef:** Numeric vector of random effects on the original scale.
- u:** Numeric vector of orthogonal, constant variance, random effects.
- eta:** The linear predictor at the current values of the parameters and the random effects.
- mu:** The means of the responses at the current parameter values.
- muEta:** The diagonal of the Jacobian of  $\mu$  by  $\eta$ . Has length zero (0) except for generalized mixed models.
- var:** The diagonal of the conditional variance of  $Y$  given the random effects, up to prior weights. In generalized mixed models this is the value of the variance function for the [glm](#) family.
- resid:** The residuals,  $y - \mu$ , weighted by the `sqrtrWt` slot (when its length is  $> 0$ ).
- sqrtrWt:** The square root of the weights applied to the model matrices  $X$  and  $Z$ . This may be of length zero (0), indicating unit weights.
- sqrtrWt:** The square root of the weights applied to the residuals to obtain the weighted residual sum of squares. This may be of length zero (0), indicating unit weights.
- RZX:** The dense solution (class "`matrix`") to  $LRZX = ST'Z'X = AX$ .
- RX:** The upper Cholesky factor (class "`matrix`") of the downdated  $X'X$ .
- The "`summary.mer`" class *contains* the "`mer`", class and has additional slots,
- methTitle:** character string specifying a method title
- logLik:** the same as `logLik(object)`.
- ngrps:** the number of levels per grouping factor in the `flist` slot.
- sigma:** the scale factor for the variance-covariance estimates
- coefs:** the matrix of estimates, standard errors, etc. for the fixed-effects coefficients
- vcov:** the same as `vcov(object)`.
- REmat:** the formatted Random-Effects matrix
- AICTab:** A named vector of values of AIC, BIC, log-likelihood and deviance

## Methods

- VarCorr** `signature(x = "mer")`: Extract variance and correlation components. See [VarCorr](#)
- anova** `signature(object = "mer")`: returns the sequential decomposition of the contributions of fixed-effects terms or, for multiple arguments, model comparison statistics. See [anova](#).

- coef** signature(object = "mer"): returns an object similar to the [ranef](#) method but incorporating the fixed-effects parameters, thereby forming a table of linear model coefficients (the columns) by level of the grouping factor (the rows).
- coerce** signature(from = "mer", to = "dtCMatrix"): returns the L slot as a "dtCMatrix" (column-oriented, sparse, triangular matrix) object.
- deviance** signature(object = "mer"): returns the [deviance](#) of the fitted model, or the "REML deviance" (i.e. negative twice the REML criterion), according to the REML argument. See the arguments section above for a description of the REML argument.
- expand** signature(object = "mer"): returns a list of terms in the expansion of the ST slot. If sparse is TRUE, the default, the elements of the list are the numeric scalar "sigma", the REML or ML estimate of the standard deviation in the model, and three sparse matrices: "P", the permutation matrix, "S", the diagonal scale matrix and "T", the lower triangular matrix determining correlations. When sparse is FALSE each element of the list is the expansions of the corresponding element of the ST slot into a list of S, the diagonal matrix, and T, the (dense) unit lower triangular matrix.
- fitted** signature(object = "mer"): returns the fitted conditional means of the responses. See [fitted](#). The [napredict](#) function is called to align the result with the original data if the model was fit with `na.action = na.exclude`.
- fixef** signature(object = "mer"): returns the estimates of the fixed-effects parameters. See [fixef](#).
- formula** signature(x = "mer"): returns the model formula. See [formula](#).
- logLik** signature(object = "mer"): returns the log-likelihood or the REML criterion, according to the optional REML argument (see the arguments section above), of the fitted model. See also [logLik](#).  
Note that [AIC](#) and [BIC](#) methods automatically work (via [logLik\(\)](#)).
- mcmcscamp** signature(object = "mer"): Create a Markov chain Monte Carlo sample from a posterior distribution of the model's parameters. See [mcmcscamp](#) for details.
- model.frame** signature(formula = "mer"): returns the model frame (the frame slot).
- model.matrix** signature(object = "mer"): returns the model matrix for the fixed-effects parameters (the X slot).
- print** signature(x = "mer"): print information about the fitted model. See the arguments section above for a description of optional arguments.
- ranef** signature(object = "mer"): returns the conditional modes of the random effects. See [ranef](#).
- resid** signature(object = "mer"): returns the (raw) residuals. This method calls [napredict](#). See the above description of the fitted method for details. See also [resid](#).
- residuals** signature(object = "mer"): Another name for the resid method.
- show** signature(object = "mer"): Same as the print method without the optional arguments.
- terms** signature(x = "mer"): Extract the [terms](#) object for the fixed-effects terms in the model formula.

- update** signature(object = "mer"): see [update](#) on how to update fitted models.
- vcov** signature(object = "mer"): Calculate variance-covariance matrix of the *fixed* effect terms, see also [vcov](#).
- with** signature(data = "mer"): Evaluate an R expression in an environment constructed from the frame slot.

### See Also

[lmer\(\)](#), [glmer\(\)](#) and [nlmer\(\)](#), which produce these objects.  
[VarCorr](#) for extracting the variance and correlation components of the *random*-effects terms.  
[mcmcSamp](#) for posterior MCMC sampling of a mer fit; [simulate-mer](#) for simulation and parametric bootstrapping

### Examples

```
(fm2 <- lmer(Reaction ~ Days + (1|Subject) + (0+Days|Subject),
            data = sleepstudy))
print(fm2, digits = 10, corr = FALSE) # more precision; no corr.matrix

logLik(fm2)
(V2 <- vcov(fm2))
terms(fm2)
str(model.matrix(fm2))
str(model.frame(fm2))
str(resid(fm2))

VarCorr(fm2)
ee <- expand(fm2)
op <- options(digits = 3)
tcrossprod(ee$sigma * ee$P %*% ee$T %*% ee$S)
options(op)
```

---

merMCMC-class

Mixed-model Markov chain Monte Carlo results

---

### Description

Objects of class "merMCMC" are Markov chain Monte Carlo samples from the distribution of the parameters of a fitted mixed-effects model.

### Objects from the Class

Objects can be created by calls of the form `new("merMCMC", ...)` or, more commonly, via the `mer` method for the generic `mcmcSamp` function.

**Slots**

- Gp**: a copy of the Gp slot of the original [mer](#) object
- ST**: matrix of samples from the parameters determining the ST slot of the [mer](#) object
- call**: Matched call from the original [mer](#) object
- deviance**: vector of samples of the (ML) deviance
- dims**: a copy of the dims slot of the original [mer](#) object
- fixef**: matrix of samples of the fixed-effects parameters
- nc**: Integer vector of length `dims["nf"]`. The number of columns of random effects in each term.
- ranef**: matrix of samples of the random effects. This matrix has zero columns unless `saveb = TRUE` is specified in the call to `mcmcscamp`. Consider the size of this matrix, which could be very large, before setting `saveb = TRUE`.
- sigma**: vector of samples of the common scale parameter or `numeric(0)` if `dims["useSc"]` is `FALSE`.

**Methods**

- HPDinterval** signature(object = "merMCMC"): use the chain to calculate Highest Posterior Density (HPD) intervals of a given empirical probability content for the model parameters. See [HPDinterval](#).
- VarCorr** signature(x = "merMCMC"): transform the ST and sigma slots to some combination of variances, covariances, standard deviations and correlations. See [VarCorr](#) for details.
- as.data.frame** signature(x = "merMCMC"): returns the fixef-effects and variance-covariance parameters from the chain in the form of a data frame. The type argument for the [VarCorr](#) method can be passed to this method to select the type of variance-covariance parameters returned.
- as.matrix** signature(x = "merMCMC"): Same as the `as.data.frame` method described above but returning a matrix.
- coerce** signature(from = "merMCMC", to = "data.frame"): Same as the `as.data.frame` method.
- densityplot** signature(object = "merMCMC"): plot empirical densities for the parameters from the chain. See also [densityplot](#).
- qqmath** signature(object = "merMCMC"): plot quantile-quantile plots for the parameters from the sample in the chain. See also [qqmath](#).
- xyplot** signature(object = "merMCMC"): plot traces of the parameter samples in the chain.

**See Also**

`mcmcscamp` produces these objects, `lmer`, `glmer` and `nlmer` produce the [mer](#) objects.

**Examples**

```
showClass("merMCMC")
```

---

 Pastes

*Paste strength by batch and cask*


---

**Description**

Paste str

**Usage**

```
data(Pastes)
```

**Format**

A data frame with 60 observations on the following 4 variables.

strength paste strength.

batch delivery batch from which the sample was sample. A factor with 10 levels: 'A' to 'J'.

cask cask within the delivery batch from which the sample was chosen. A factor with 3 levels: 'a' to 'c'.

sample the sample of paste whose strength was assayed, two assays per sample. A factor with 30 levels: 'A:a' to 'J:c'.

**Details**

The data are described in Davies and Goldsmith (1972) as coming from “ deliveries of a chemical paste product contained in casks where, in addition to sampling and testing errors, there are variations in quality between deliveries ... As a routine, three casks selected at random from each delivery were sampled and the samples were kept for reference. ... Ten of the delivery batches were sampled at random and two analytical tests carried out on each of the 30 samples”.

**Source**

O.L. Davies and P.L. Goldsmith (eds), *Statistical Methods in Research and Production, 4th ed.*, Oliver and Boyd, (1972), section 6.5

**Examples**

```
str(Pastes)
dotplot(cask ~ strength | reorder(batch, strength), Pastes,
        strip = FALSE, strip.left = TRUE, layout = c(1, 10),
        ylab = "Cask within batch",
        xlab = "Paste strength", jitter.y = TRUE)
## Modifying the factors to enhance the plot
Pastes <- within(Pastes, batch <- reorder(batch, strength))
Pastes <- within(Pastes, sample <- reorder(reorder(sample, strength),
        as.numeric(batch)))
dotplot(sample ~ strength | batch, Pastes,
        strip = FALSE, strip.left = TRUE, layout = c(1, 10),
```

```

scales = list(y = list(relation = "free")),
ylab = "Sample within batch",
xlab = "Paste strength", jitter.y = TRUE)
## Four equivalent models differing only in specification
(fm1 <- lmer(strength ~ (1|batch) + (1|sample), Pastes))
(fm2 <- lmer(strength ~ (1|batch/cask), Pastes))
(fm3 <- lmer(strength ~ (1|batch) + (1|batch:cask), Pastes))
(fm4 <- lmer(strength ~ (1|batch/sample), Pastes))
## fm4 results in redundant labels on the sample:batch interaction
head(ranef(fm4)[[1]])
## compare to fm1
head(ranef(fm1)[[1]])
## This model is different and NOT appropriate for these data
(fm5 <- lmer(strength ~ (1|batch) + (1|cask), Pastes))
image(fm1@L, sub = "Structure of random effects interaction in pastes model")

```

---

 Penicillin

*Variation in penicillin testing*


---

### Description

Six samples of penicillin were tested using the *B. subtilis* plate method on each of 24 plates. The response is the diameter (mm) of the zone of inhibition of growth of the organism.

### Usage

```
data(Penicillin)
```

### Format

A data frame with 144 observations on the following 3 variables.

diameter diameter (mm) of the zone of inhibition of the growth of the organism.

plate assay plate. A factor with levels 'a' to 'x'.

sample penicillin sample. A factor with levels 'A' to 'F'.

### Details

The data are described in Davies and Goldsmith (1972) as coming from an investigation to “assess the variability between samples of penicillin by the *B. subtilis* method. In this test method a bulk-inoculated nutrient agar medium is poured into a Petri dish of approximately 90 mm. diameter, known as a plate. When the medium has set, six small hollow cylinders or pots (about 4 mm. in diameter) are cemented onto the surface at equally spaced intervals. A few drops of the penicillin solutions to be compared are placed in the respective cylinders, and the whole plate is placed in an incubator for a given time. Penicillin diffuses from the pots into the agar, and this produces a clear circular zone of inhibition of growth of the organisms, which can be readily measured. The diameter of the zone is related in a known way to the concentration of penicillin in the solution.”

**Source**

O.L. Davies and P.L. Goldsmith (eds), *Statistical Methods in Research and Production, 4th ed.*, Oliver and Boyd, (1972), section 6.6

**Examples**

```
str(Penicillin)
dotplot(reorder(plate, diameter) ~ diameter, Penicillin, groups = sample,
        ylab = "Plate", xlab = "Diameter of growth inhibition zone (mm)",
        type = c("p", "a"), auto.key = list(columns = 3, lines = TRUE,
        title = "Penicillin sample"))
(fm1 <- lmer(diameter ~ (1|plate) + (1|sample), Penicillin))
image(fm1@L,
      sub = "Structure of random effects interaction in penicillin model")
```

---

 ranef

---

*Extract the modes of the random effects*


---

**Description**

A generic function to extract the conditional modes of the random effects from a fitted model object. For linear mixed models the conditional modes of the random effects are also the conditional means.

**Usage**

```
ranef(object, ...)
## S4 method for signature 'mer'
ranef(object, postVar = FALSE, drop = FALSE,
      which1 = names(wt), ...)
```

**Arguments**

object	an object of a class of fitted models with random effects, typically an <i>"mer"</i> object.
postVar	an optional logical argument indicating if the conditional variance-covariance matrices, also called the "posterior variances", of the random effects should be added as an attribute. Default is FALSE.
drop	an optional logical argument indicating components of the return value that would be data frames with a single column, usually a column called '(Intercept)', should be returned as named vectors.
which1	character vector of names of factors for which to return results.
...	some methods for this generic function require additional arguments.

## Details

If grouping factor  $i$  has  $k$  levels and  $j$  random effects per level the  $i$ th component of the list returned by `ranef` is a data frame with  $k$  rows and  $j$  columns. If `postVar` is `TRUE` the "postVar" attribute is an array of dimension  $j$  by  $j$  by  $k$ . The  $k$ th face of this array is a positive definite symmetric  $j$  by  $j$  matrix. If there is only one grouping factor in the model the variance-covariance matrix for the entire random effects vector, conditional on the estimates of the model parameters and on the data will be block diagonal and this  $j$  by  $j$  matrix is the  $k$ th diagonal block. With multiple grouping factors the faces of the "postVar" attributes are still the diagonal blocks of this conditional variance-covariance matrix but the matrix itself is no longer block diagonal.

## Value

A list of data frames, one for each grouping factor for the random effects. The number of rows in the data frame is the number of levels of the grouping factor. The number of columns is the dimension of the random effect associated with each level of the factor.

If `postVar` is `TRUE` each of the data frames has an attribute called "postVar" which is a three-dimensional array with symmetric faces.

When `drop` is `TRUE` any components that would be data frames of a single column are converted to named numeric vectors.

## Note

To produce a "caterpillar plot" of the random effects apply `dotplot` to the result of a call to `ranef` with `postVar = TRUE`.

## Examples

```
fm1 <- lmer(Reaction ~ Days + (Days|Subject), sleepstudy)
fm2 <- lmer(Reaction ~ Days + (1|Subject) + (0+Days|Subject), sleepstudy)
fm3 <- lmer(diameter ~ (1|plate) + (1|sample), Penicillin)
ranef(fm1)
str(rr1 <- ranef(fm1, postVar = TRUE))
dotplot(rr1, scales = list(x = list(relation = 'free')))[["Subject"]]
##str(ranef(fm2, postVar = TRUE)) ## code not yet written
op <- options(digits = 4)
ranef(fm3, drop = TRUE)
options(op)
```

---

 refit

*Re-fit a model to a new response vector*


---

## Description

This generic function fits a model to a new response vector. It typically is used as part of a simulation. For models fit by `lmer` or `nlmer` it is much faster to use this function to re-fit the model than to begin from the model formula.

**Usage**

```
refit(object, newresp, ...)
```

**Arguments**

object	a fitted model. Methods are defined for models fit by <code>lmer</code> and by <code>nlmer</code> .
newresp	a new response vector - typically a numeric vector.
...	further arguments passed for some methods.

**Details**

For many model-fitting functions in R a large part of the execution time is taken up with converting a formula representation and the data argument into the numerical structures used to evaluate the parameter estimates. It is common in a simulation to use the same model specification and covariate data on many simulated responses. That is, the only thing that changes between model fits is the response vector. This generic function allows for the model specification to and covariate data to stay the same while the only the response vector is changed.

**Value**

a fitted model similar to object

**Examples**

```
fm1 <- lmer(Reaction ~ Days + (Days|Subject), sleepstudy)
set.seed(54321)
system.time(simfe <-
  data.frame(t(apply(simulate(fm1, nsim = 100),
    2, function(y) fixef(refit(fm1, y)))),
    check.names = FALSE))
qqmath(~ '(Intercept)' + Days, simfe, outer = TRUE,
  ylab = NULL, scales = list(y = list(relation = "free")),
  layout = c(1,2), aspect = 1, type = c("g", "p"))
sapply(simfe, mean)
sapply(simfe, sd)
```

---

simulate-mer

*Simulate based on mer fits*

---

**Description**

These generic functions (1) generate simulations based on the estimated fitted models (conditional on the estimated values of both the random and fixed effects) and (2) efficiently refit a specified model based on a new vector, data frame, or (for binomial models) matrix of responses

**Usage**

```
## S4 method for signature 'mer'
simulate(object, nsim = 1 , seed = NULL, ...)
## S4 method for signature 'mer'
refit(object, newresp, ...)
```

**Arguments**

object	An object of a suitable class - usually an "mer" object.
nsim	(integer) number of simulations to generate
seed	(integer or NULL) an object specifying if and how the random number generator should be initialized ('seeded'). If an integer, seed is used in a call to <code>set.seed</code> before simulating the response vectors. If set, the value is saved as the 'seed' attribute of the returned value. The default, 'NULL', will not change the random generator state, and return ' <code>Random.seed</code> ' as the 'seed' attribute
newresp	a numeric vector or single-column data frame (for most models) or a matrix or two-column data frame (for binomial models where the response was specified as a matrix) containing new values for the response variable
...	Some methods may take additional, optional arguments.

**Value**

Data frame representing one or more simulations from the fitted model, conditional on both the fixed and random effect estimates. In cases of univariate responses (i.e. all cases except binomial models where the response is specified as a two-column matrix), each column of the data frame represents an independent simulation.

For binomial models where the response is specified as a two-column matrix, the value is a non-standard data frame where each column of the data frame (as accessed via `x[,i]` or `x[[i]]`) is actually a two-column *matrix* of responses (successes and failures).

**Methods**

**refit** signature(object = "mer", newresp = "numeric"): Update the response vector only and refit the model. See [refit](#).

**simulate** signature(object = "mer"): simulate nsim (defaults to 1) responses from the theoretical distribution corresponding to the fitted model. The `refit` method is particularly useful in combination with this method. See also [simulate](#).

**Examples**

```
fm1 <- lmer(Reaction ~ Days + (Days|Subject), sleepstudy)
fm2 <- lmer(Reaction ~ Days + (1|Subject) + (0+Days|Subject), sleepstudy)

## generic parametric bootstrapping function; return a single simulated deviance
## difference between full (m1) and reduced (m0) models under the
## null hypothesis that the reduced model is the true model
pboot <- function(m0,m1) {
  s <- simulate(m0)
```

```

L0 <- logLik(refit(m0,s))
L1 <- logLik(refit(m1,s))
2*(L1-L0)
}

obsdev <- c(2*(logLik(fm1)-logLik(fm2)))
## Not run:
## parametric bootstrap test of significance of correlation between
## random effects of '(Intercept)' and Days
## Timing: approx. 70 secs on a 2.66 GHz Intel Core Duo laptop
set.seed(1001)
sleepstudy_PB <- replicate(500,pboot(fm2,fm1))

## End(Not run)
library(lattice)
## null value for correlation parameter is *not* on the boundary
## of its feasible space, so we would expect chisq(1)
qqmath(sleepstudy_PB,distribution=function(p) qchisq(p,df=1),
       type="l",
       prepanel = prepanel.qqmathline,
       panel = function(x, ...) {
         panel.qqmathline(x, ...)
         panel.qqmath(x, ...)
       })
## classical test
pchisq(obsdev,df=1,lower.tail=FALSE)
## parametric bootstrap-based test
mean(sleepstudy_PB>obsdev)
## pretty close in this case

## cbpp data
## PB test of significance of main effect of period
gm1 <- glmer(cbind(incidence, size - incidence) ~ period + (1 | herd),
            family = binomial, data = cbpp)
gm0 <- update(gm1, . ~. -period)

## Not run:
cbpp_PB <- replicate(500,pboot(gm0,gm1))

## End(Not run)

obsdev <- c(2*(logLik(gm1)-logLik(gm0)))
qqmath(cbpp_PB,distribution=function(p) qchisq(p,df=3),
       type="l",
       prepanel = prepanel.qqmathline,
       panel = function(x, ...) {
         panel.qqmathline(x, ...)
         panel.qqmath(x, ...)
       })
## classical test
pchisq(obsdev,df=3,lower.tail=FALSE)
## parametric bootstrap-based test
mean(cbpp_PB>obsdev)

```

```
## alternative plot
nsim <- length(cbpp_PB)
pdat <- data.frame(PB=(1:nsim)/(nsim+1),
  nominal=pchisq(sort(cbpp_PB,decreasing=TRUE),3,lower.tail=FALSE))
xyplot(nominal~PB,data=pdat,type="l",grid=TRUE,
  scales=list(x=list(log=TRUE),y=list(log=TRUE)),
  panel=function(...) {
    panel.abline(a=0,b=1,col="darkgray")
    panel.xyplot(...)
  })
```

---

sleepstudy

*Reaction times in a sleep deprivation study*

---

## Description

The average reaction time per day for subjects in a sleep deprivation study. On day 0 the subjects had their normal amount of sleep. Starting that night they were restricted to 3 hours of sleep per night. The observations represent the average reaction time on a series of tests given each day to each subject.

## Usage

```
data(sleepstudy)
```

## Format

A data frame with 180 observations on the following 3 variables.

Reaction Average reaction time (ms)

Days Number of days of sleep deprivation

Subject Subject number on which the observation was made.

## Details

These data are from the study described in Belenky et al. (2003), for the sleep-deprived group and for the first 10 days of the study, up to the recovery period.

## References

Gregory Belenky, Nancy J. Wesensten, David R. Thorne, Maria L. Thomas, Helen C. Sing, Daniel P. Redmond, Michael B. Russo and Thomas J. Balkin (2003) Patterns of performance degradation and restoration during sleep restriction and subsequent recovery: a sleep dose-response study. *Journal of Sleep Research* **12**, 1–12.

**Examples**

```
str(sleepstudy)
xyplot(Reaction ~ Days | Subject, sleepstudy, type = c("g","p","r"),
       index = function(x,y) coef(lm(y ~ x))[1],
       xlab = "Days of sleep deprivation",
       ylab = "Average reaction time (ms)", aspect = "xy")
(fm1 <- lmer(Reaction ~ Days + (Days|Subject), sleepstudy))
(fm2 <- lmer(Reaction ~ Days + (1|Subject) + (0+Days|Subject), sleepstudy))
```

---

sleepstudy\_PB

*Stored parametric bootstrap samples for the sleepstudy data*


---

**Description**

Parametric bootstrap distribution computed from models fitted to the [sleepstudy](#) data set

**Usage**

```
sleepstudy_PB
```

**Format**

Contains a vector of 500 computed deviances under the (simulated) null hypothesis

**See Also**

[simulate-mer](#) for examples of what to do with the parametric bootstrap results

**Examples**

```
fm1 <- lmer(Reaction ~ Days + (Days|Subject), sleepstudy)
fm2 <- lmer(Reaction ~ Days + (1|Subject) + (0+Days|Subject), sleepstudy)

## generic parametric bootstrapping function; return a single simulated deviance
## difference between full ('m1') and reduced ('m0') models under the
## null hypothesis that the reduced model is the true model
pboot <- function(m0,m1) {
  s <- simulate(m0)
  L0 <- logLik(refit(m0,s))
  L1 <- logLik(refit(m1,s))
  2*(L1-L0)
}

obsdev <- c(2*(logLik(fm1)-logLik(fm2)))
## Not run:
## parametric bootstrap test of significance of correlation between
## random effects of '(Intercept)' and Days
## Timing: approx. 70 secs on a 2.66 GHz Intel Core Duo laptop
set.seed(1001)
```

```
sleepstudy_PB <- replicate(500, pboot(fm2, fm1))

## End(Not run)
```

---

 VarCorr

*Extract variance and correlation components*


---

### Description

Extract the estimated variances, standard deviations, and correlations of the random-effects terms in a mixed-effects model, of class `mer`.

When appropriate, the within-group error variance and standard deviation are also calculated.

### Usage

```
## S4 method for signature 'mer'
VarCorr(x, ...)
## S4 method for signature 'merMCMC'
VarCorr(x, type = c("raw", "varcov", "sdcorr", "logs"), ...)
```

### Arguments

<code>x</code>	a fitted model object, usually an object inheriting from class <code>mer</code> .
<code>type</code>	character string indicating the type of result to be returned, either "raw", the raw representation as in the ST slot of the <code>mer</code> class, or "varcov", variances and covariances, or "sdcorr", standard deviations and correlations, or "logs", logs of the standard deviations and Fisher's z transformation of the correlations.
<code>...</code>	Additional, optional arguments for some methods. At present none are used.

### Value

an object of class `VarCorr`.

### See Also

the `lmer` function and `mer` class; the result class `VarCorr`.

### Examples

```
(fm2 <- lmer(Reaction ~ Days + (1|Subject) + (0+Days|Subject),
            data = sleepstudy))
(VC <- VarCorr(fm2))
```

---

VarCorr-class	<i>Class "VarCorr"</i>
---------------	------------------------

---

**Description**

The variance and correlation information for a random-effects model.

**Objects from the Class**

Objects can be created by calls of the form `new("VarCorr", ...)`.

**Slots**

**scale:** The estimated standard deviation of the lowest level noise term in the model.

**reSumry:** Object of class "list", a list containing the summary of the positive-definite matrices for the random-effects levels.

**Methods**

**show** signature(object = "VarCorr"): show the object.

---

VerbAgg	<i>Verbal Aggression item responses</i>
---------	---

---

**Description**

These are the item responses to a questionnaire on verbal aggression. These data are used throughout De Boeck and Wilson, *Explanatory Item Response Models* (Springer, 2004) to illustrate various forms of item response models.

**Format**

A data frame with 7584 observations on the following 13 variables.

**Anger** the subject's Trait Anger score as measured on the State-Trait Anger Expression Inventory (STAXI)

**Gender** the subject's gender - a factor with levels M and F

**item** the item on the questionnaire, as a factor

**resp** the subject's response to the item - an ordered factor with levels no < perhaps < yes

**id** the subject identifier, as a factor

**btype** behavior type - a factor with levels curse, scold and shout

**situ** situation type - a factor with levels other and self indicating other-to-blame and self-to-blame

**mode** behavior mode - a factor with levels want and do

**r2** dichotomous version of the response - a factor with levels N and Y

**Source**

<http://bear.soe.berkeley.edu/EIRM/>

**References**

De Boeck and Wilson (2004), *Explanatory Item Response Models*, Springer.

**Examples**

```
str(VerbAgg)
xtabs(~ item + resp, VerbAgg)
xtabs(~ btype + resp, VerbAgg)
round(100 * ftable(prop.table(xtabs(~ situ + mode + resp, VerbAgg), 1:2), 1))
person <- unique(subset(VerbAgg, select = c(id, Gender, Anger)))
if (require(lattice)) {
  densityplot(~ Anger, person, groups = Gender, auto.key = list(columns = 2),
             xlab = "Trait Anger score (STAXI)")
}
```

# Index

## \*Topic **classes**

- lmlist-class, 12
- mer-class, 14
- merMCMC-class, 18
- VarCorr-class, 30

## \*Topic **datagen**

- mcmcsamp, 13
- simulate-mer, 24

## \*Topic **datasets**

- cake, 2
- cbpp, 3
- cbpp\_PB, 4
- Dyestuff, 5
- Pastes, 20
- Penicillin, 21
- sleepstudy, 27
- sleepstudy\_PB, 28
- VerbAgg, 30

## \*Topic **htest**

- HPDinterval, 8

## \*Topic **methods**

- lmer, 9
- mcmcsamp, 13
- ranef, 22
- refit, 23
- simulate-mer, 24

## \*Topic **models**

- fixef, 6
- lmer, 9
- lmlist, 11
- ranef, 22
- refit, 23
- VarCorr, 29

## \*Topic **univar**

- HPDinterval, 8

## \*Topic **utilities**

- getME, 7

AIC, 17

anova, 16

anova, mer-method (mer-class), 14

as.data.frame, merMCMC-method  
(merMCMC-class), 18

as.matrix, merMCMC-method  
(merMCMC-class), 18

cake, 2

cbpp, 3, 4

cbpp\_PB, 4

coef, lmlist-method (lmlist-class), 12

coef, mer-method (mer-class), 14

coef, summary.mer-method (mer-class), 14

coerce, mer, dtCMatrix-method  
(mer-class), 14

coerce, merMCMC, data.frame-method  
(merMCMC-class), 18

coerce, merMCMC, matrix-method  
(merMCMC-class), 18

confint, lmlist-method (lmlist-class), 12

dCHMfactor, 16

densityplot, 19

densityplot, merMCMC-method  
(merMCMC-class), 18

deviance, 17

deviance, mer-method (mer-class), 14

deviance, summary.mer-method  
(mer-class), 14

dgCMatrix, 15, 16

dotplot, 23

Dyestuff, 5

Dyestuff2 (Dyestuff), 5

expand, mer-method (mer-class), 14

family, 9

fitted, 17

fitted, mer-method (mer-class), 14

fixed.effects (fixef), 6

fixef, 6, 7, 17

- fixef, ANY-method (fixef), 6
- fixef, mer-method (fixef), 6
- formula, 17
- formula, lmlist-method (lmlist-class), 12
- formula, mer-method (mer-class), 14
  
- gaussian, 10
- getME, 7
- glm, 9, 16
- glmer, 7, 15, 18, 19
- glmer (lmer), 9
  
- HPDinterval, 8, 19
- HPDinterval, matrix-method (HPDinterval), 8
- HPDinterval, merMCMC-method (HPDinterval), 8
  
- lm, 10–12
- lmer, 7, 9, 15, 18, 19, 23, 24, 29
- lmer2 (lmer), 9
- lmlist, 11, 12
- lmlist, formula, data.frame-method (lmlist), 11
- lmlist-class, 12
- logLik, 16, 17
- logLik, mer-method (mer-class), 14
- logLik, summary.mer-method (mer-class), 14
  
- mcmcsamp, 13, 17–19
- mcmcsamp, mer-method (mcmcsamp), 13
- mer, 7, 11, 13, 19, 22, 25, 29
- mer-class, 14
- merMCMC, 8, 14
- merMCMC-class, 18
- model.frame, mer-method (mer-class), 14
- model.matrix, mer-method (mer-class), 14
  
- na.fail, 12
- na.omit, 12
- name, 7
- napredict, 17
- nlmer, 7, 15, 18, 19, 23, 24
- nlmer (lmer), 9
  
- options, 12
  
- Pastes, 20
- Penicillin, 21
  
- plot, 13
- plot, lmlist.confint, ANY-method (lmlist-class), 12
- print, mer-method (mer-class), 14
  
- qqmath, 19
- qqmath, merMCMC-method (merMCMC-class), 18
  
- ranef, 7, 17, 22
- ranef, mer-method (ranef), 22
- ranef-methods (ranef), 22
- refit, 23, 25
- refit, mer, data.frame-method (simulate-mer), 24
- refit, mer, matrix-method (refit), 23
- refit, mer, numeric-method (refit), 23
- refit, mer-method (simulate-mer), 24
- resid, 17
- resid, mer-method (mer-class), 14
- residuals, mer-method (mer-class), 14
  
- show, 13, 30
- show, lmlist-method (lmlist-class), 12
- show, mer-method (mer-class), 14
- show, VarCorr-method (VarCorr-class), 30
- simulate, 25
- simulate, mer-method (simulate-mer), 24
- simulate-mer, 4, 18, 28
- simulate-mer, 24
- sleepstudy, 27, 28
- sleepstudy\_PB, 28
- slot, 7
- summary, mer-method (mer-class), 14
- summary, summary.mer-method (mer-class), 14
- summary.mer-class (mer-class), 14
  
- terms, 17
- terms, mer-method (mer-class), 14
  
- update, 13, 18
- update, lmlist-method (lmlist-class), 12
- update, mer-method (mer-class), 14
  
- VarCorr, 16, 18, 19, 29, 29
- VarCorr, mer-method (VarCorr), 29
- VarCorr, merMCMC-method (VarCorr), 29
- VarCorr-class, 30
- vcov, 7, 16, 18

`vcov`, `mer-method` (`mer-class`), 14  
`vcov`, `summary.mer-method` (`mer-class`), 14  
`VerbAgg`, 30

`with`, `mer-method` (`mer-class`), 14

`xyplot`, `merMCMC-method` (`merMCMC-class`),  
18