

Package ‘buildmer’

September 3, 2019

Title Stepwise Elimination and Term Reordering for Mixed-Effects Regression

Version 1.2.1

Description Finds the largest possible regression model that will still converge for various types of regression analyses (including mixed models and generalized additive models) and then optionally performs stepwise elimination similar to the forward and backward effect selection methods in SAS, based on the change in log-likelihood or its significance, Akaike's Information Criterion, or the Bayesian Information Criterion.

Depends R (>= 3.2)

Imports methods, mgcv, lme4, plyr, stats, utils

Suggests JuliaCall, MASS, gamm4, glmertree, glmmTMB, knitr, lmerTest, nlme, nnet, parallel, pbkrtest, rmarkdown

License FreeBSD

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

BugReports <https://github.com/cvoeten/buildmer/issues>

VignetteBuilder knitr

NeedsCompilation no

Author Cesko C. Voeten [aut, cre]

Maintainer Cesko C. Voeten <cvoeten@gmail.com>

Repository CRAN

Date/Publication 2019-09-03 13:40:10 UTC

R topics documented:

buildmer-package	2
add.terms	4
build.formula	5

buildbam	6
buildcustom	7
buildgam	8
buildgamm4	9
buildglmmTMB	10
buildgls	11
buildjulia	12
buildlme	14
buildmer	15
buildmer-class	16
buildmertree	16
buildmultinom	17
conv	18
diag.formula-method	19
migrant	20
remove.terms	20
tabulate.formula	21
vowels	21

Index 22

buildmer-package	<i>Construct and fit as complete a model as possible and perform step-wise elimination</i>
------------------	--

Description

The `buildmer` package consists of a number of functions, each designed to fit specific types of models (e.g. `buildmer` for mixed-effects regression, `buildgam` for generalized additive models, `buildmertree` for mixed-effects-regression trees, and so forth. The common parameters shared by all (or most of) these functions are documented here. If you are looking for a more general description of what the various `build...` functions do, see under ‘Details’. For function-specific details, see the documentation for each individual function.

Arguments

formula	The model formula for the maximal model you would like to fit. Alternatively, a <code>buildmer</code> term list as obtained from <code>tabulate.formula</code> . In the latter formulation, you also need to specify a <code>dep=...</code> argument specifying the dependent variable to go along with the term list. See <code>tabulate.formula</code> for an example of where this is useful
data	The data to fit the model(s) to
family	The error distribution to use
cl	An optional cluster object as returned by function <code>makeCluster</code> from package <code>parallel</code> to use for parallelizing the evaluation of terms. Note that, if and only if using the <code>cl</code> functionality, the data and other arguments will be searched for in the global environment only, so you should manually set up the cluster’s

environments using `clusterExport()` if necessary. In addition, some buildmer-internal objects will be exported to the cluster nodes. These will be cleaned up afterwards, but any already-present objects with the same name (e.g. 'p' will be overwritten)

direction	Character string or vector indicating the direction for stepwise elimination; possible options are 'order' (order terms by their contribution to the model), 'backward' (backward elimination), 'forward' (forward elimination, implies order). The default is the combination <code>c('order', 'backward')</code> , to first make sure that the model converges and to then perform backward elimination; other such combinations are perfectly allowed
crit	Character string or vector determining the criterion used to test terms for elimination. Possible options are 'LRT' (likelihood-ratio test; this is the default), 'LL' (use the raw $-2 \log$ likelihood), 'AIC' (Akaike Information Criterion), and 'BIC' (Bayesian Information Criterion)
include	A one-sided formula or character vector of terms that will be kept in the model at all times. These do not need to be specified separately in the <code>formula</code> argument. Useful for e.g. passing correlation structures in <code>glmmTMB</code> models
reduce.fixed	Logical indicating whether to reduce the fixed-effect structure
reduce.random	Logical indicating whether to reduce the random-effect structure
calc.anova	Logical indicating whether to also calculate the ANOVA table for the final model after term elimination
calc.summary	Logical indicating whether to also calculate the summary table for the final model after term elimination

Details

With the default options, all buildmer functions will do two things:

1. Determine the order of the effects in your model, based on their importance as measured by the likelihood-ratio test statistic. This identifies the 'maximal model', which is the model containing either all effects specified by the user, or subset of those effects that still allow the model to converge, ordered such that the most information-rich effects have made it in.
2. Perform backward stepwise elimination based on the significance of the change in log-likelihood.

The final model is returned in the `model` slot of the returned buildmer object. All functions in the buildmer package are aware of the distinction between (f)REML and ML, and know to divide chi-square p -values by 2 when comparing models differing only in random effects (see Pinheiro & Bates 2000). The steps executed above can be changed using the `direction` argument, allowing for arbitrary chains of, for instance, forward-backward-forward stepwise elimination (although using more than one elimination method on the same data is not recommended). The criterion for determining the importance of terms in the ordering stage and the elimination of terms in the elimination stage can also be changed, using the `crit` argument.

Examples

```
# Only finding the maximal model, with importance of effects measured by AIC, parallelizing the
```

```

# model evaluations using two cores, using the bobyqa optimizer and asking for verbose output
library(parallel)
cl <- makeCluster(2,outfile='')
control <- lme4::lmerControl(optimizer='bobyqa')
clusterExport(cl,'control') #this is not done automatically for '...' arguments!
m <- buildmer(f1 ~ vowel*timepoint*following + (vowel*timepoint*following|participant) +
              (timepoint|word),data=vowels,cl=cl,direction='order',crit='AIC',calc.anova=FALSE,
              calc.summary=FALSE,control=control,verbose=2)
# The maximal model is: f1 ~ vowel + timepoint + vowel:timepoint + following +
# timepoint:following +vowel:following + vowel:timepoint:following + (1 + timepoint +
# following + timepoint:following | participant) + (1 + timepoint | word)
# Now do backward stepwise elimination (result: f1 ~ vowel + timepoint + vowel:timepoint +
# following + timepoint:following + (1 + timepoint + following + timepoint:following |
# participant) + (1 + timepoint | word))
buildmer(formula(m@model),data=vowels,direction='backward',crit='AIC',control=control)
# Or forward (result: retains the full model)
buildmer(formula(m@model),data=vowels,direction='forward',crit='AIC',control=control)
# Print summary with p-values based on Satterthwaite denominator degrees of freedom
summary(m,ddf='Satterthwaite')

# Example for fitting a model without correlations in the random part
# (even for factor variables!)
# 1. Create explicit columns for factor variables
library(buildmer)
vowels <- cbind(vowels,model.matrix(~vowel,vowels))
# 2. Create formula with diagonal covariance structure
form <- diag(f1 ~ (vowel1+vowel2+vowel3+vowel4)*timepoint*following +
             ((vowel1+vowel2+vowel3+vowel4)*timepoint*following | participant) +
             (timepoint | word))
# 3. Convert formula to buildmer terms list, grouping terms starting with 'vowel'
terms <- tabulate.formula(form,group='vowel[^:]')
# 4. Directly pass the terms object to buildmer(), using the hidden 'dep' argument to specify
# the dependent variable
m <- buildmer(terms,data=vowels,dep='f1')

```

add.terms

Add terms to a formula

Description

Add terms to a formula

Usage

```
add.terms(formula, add)
```

Arguments

formula	The formula to add terms to.
add	A vector of terms to add. To add terms nested in random-effect groups, use ‘(term group)’ syntax if you want to add an independent random effect (e.g. ‘(olderterm group) + (term group)’), or use ‘term group’ syntax if you want to add a dependent random effect to a pre-existing term group (if no such group exists, it will be created at the end of the formula).

Value

The updated formula.

Examples

```
library(buildmer)
form <- Reaction ~ Days + (1|Subject)
add.terms(form, 'Days|Subject')
add.terms(form, '(0+Days|Subject)')
add.terms(form, c('many', 'more|terms', 'to|terms', '(be|added)', 'to|test'))
```

build.formula	<i>Convert a buildmer term list into a proper model formula</i>
---------------	---

Description

Convert a buildmer term list into a proper model formula

Usage

```
build.formula(dep, terms, env = parent.frame())
```

Arguments

dep	The dependent variable.
terms	The term list.
env	The environment of the formula to return.

Value

A formula.

Examples

```

library(buildmer)
form1 <- Reaction ~ Days + (Days|Subject)
terms <- tabulate.formula(form1)
form2 <- build.formula(dep='Reaction', terms)

# check that the two formulas give the same results
library(lme4)
check <- function (f) resid(lmer(f,sleepstudy))
all.equal(check(form1),check(form2))

```

buildbam	<i>Use buildmer to fit big generalized additive models using bam from package mgcv</i>
----------	--

Description

Use buildmer to fit big generalized additive models using bam from package mgcv

Usage

```

buildbam(formula, data = NULL, family = gaussian(), cl = NULL,
  direction = c("order", "backward"), crit = "LRT", include = NULL,
  calc.anova = TRUE, calc.summary = TRUE, ...)

```

Arguments

formula	See the general documentation under buildmer-package
data	See the general documentation under buildmer-package
family	See the general documentation under buildmer-package
cl	See the general documentation under buildmer-package
direction	See the general documentation under buildmer-package
crit	See the general documentation under buildmer-package
include	See the general documentation under buildmer-package
calc.anova	See the general documentation under buildmer-package
calc.summary	See the general documentation under buildmer-package
...	Additional options to be passed to bam

See Also

[buildmer-package](#)

Examples

```
library(buildmer)
m <- buildbam(f1 ~ s(timepoint,by=following) + s(participant,by=following,bs='re') +
              s(participant,timepoint,by=following,bs='fs'),data=vowels)
```

buildcustom	<i>Use buildmer to perform stepwise elimination using a custom fitting function</i>
-------------	---

Description

Use buildmer to perform stepwise elimination using a custom fitting function

Usage

```
buildcustom(formula, data = NULL, cl = NULL, direction = c("order",
  "backward"), crit = function(ref, alt) stop("'crit' not specified"),
  include = NULL, reduce.fixed = TRUE, reduce.random = TRUE,
  fit = function(p, formula) stop("'fit' not specified"),
  elim = function(x) stop("'elim' not specified"), ...)
```

Arguments

formula	See the general documentation under buildmer-package
data	See the general documentation under buildmer-package
cl	See the general documentation under buildmer-package
direction	See the general documentation under buildmer-package
crit	See the general documentation under buildmer-package
include	See the general documentation under buildmer-package
reduce.fixed	See the general documentation under buildmer-package
reduce.random	See the general documentation under buildmer-package
fit	A function taking two arguments, of which the first is the buildmer parameter list p and the second one is a formula. The function must return a single object, which is treated as a model object fitted via the provided formula. The function must return an error ('stop()') if the model does not converge
elim	A function taking one argument and returning a single value. The first argument is the return value of the function passed in crit, and the returned value must be a logical indicating if the small model must be selected (return TRUE) or the large model (return FALSE)
...	Additional options to be passed to the fitting function, such as perhaps a data argument

See Also

[buildmer-package](#)

Examples

```
## Use \code{buildmer} to do stepwise linear discriminant analysis
library(buildmer)
migrant[, -1] <- scale(migrant[, -1])
flipfit <- function (p, formula) {
  # The predictors must be entered as dependent variables in a MANOVA
  # (i.e. the predictors must be flipped with the dependent variable)
  Y <- model.matrix(formula, migrant)
  m <- lm(Y ~ 0+migrant$changed)
  # the model may error out when asking for the MANOVA
  test <- try(anova(m))
  if (inherits(test, 'try-error')) test else m
}
crit.F <- function (ma, mb) { # use whole-model F
  pvals <- anova(mb)$'Pr(>F)' # not valid for backward!
  pvals[length(pvals)-1]
}
crit.Wilks <- function (ma, mb) {
  if (is.null(ma)) return(crit.F(ma, mb)) #not completely correct, but close as F approximates X2
  Lambda <- anova(mb, test='Wilks')$Wilks[1]
  p <- length(coef(mb))
  n <- 1
  m <- nrow(migrant)
  Bartlett <- ((p-n+1)/2-m)*log(Lambda)
  pchisq(Bartlett, n*p, lower.tail=FALSE)
}

# First, order the terms based on Wilks' Lambda
m <- buildcustom(changed ~ friends.nl+friends.be+multilingual+standard+hearing+reading+attention+
sleep+gender+handedness+diglossic+age+years, direction='order', fit=flipfit, crit=crit.Wilks)
# Now, use the six most important terms (arbitrary choice) in the LDA
library(MASS)
m <- lda(changed ~ diglossic + age + reading + friends.be + years + multilingual, data=migrant)
```

buildgam

Use buildmer to fit generalized additive models using gam from package mgcv

Description

Use buildmer to fit generalized additive models using gam from package mgcv

Usage

```
buildgam(formula, data = NULL, family = gaussian(), cl = NULL,
direction = c("order", "backward"), crit = "LRT", include = NULL,
calc.anova = TRUE, calc.summary = TRUE, ...)
```


Arguments

formula	See the general documentation under buildmer-package
data	See the general documentation under buildmer-package
family	See the general documentation under buildmer-package
cl	See the general documentation under buildmer-package
direction	See the general documentation under buildmer-package
crit	See the general documentation under buildmer-package
include	See the general documentation under buildmer-package
calc.anova	See the general documentation under buildmer-package
calc.summary	See the general documentation under buildmer-package
...	Additional options to be passed to bam

See Also

[buildmer-package](#)

Examples

```
library(buildmer)
m <- buildgam(f1 ~ s(timepoint,by=following) + s(participant,by=following,bs='re') +
              s(participant,timepoint,by=following,bs='fs'),data=vowels)
```

buildgamm4

Use buildmer to fit generalized additive models using package gamm4

Description

Use buildmer to fit generalized additive models using package gamm4

Usage

```
buildgamm4(formula, data = NULL, family = gaussian(), cl = NULL,
            direction = c("order", "backward"), crit = "LRT", include = NULL,
            reduce.fixed = TRUE, reduce.random = TRUE, calc.anova = TRUE,
            calc.summary = TRUE, ddf = "Wald", ...)
```

Arguments

formula	See the general documentation under buildmer-package
data	See the general documentation under buildmer-package
family	See the general documentation under buildmer-package
cl	See the general documentation under buildmer-package
direction	See the general documentation under buildmer-package
crit	See the general documentation under buildmer-package
include	See the general documentation under buildmer-package
reduce.fixed	See the general documentation under buildmer-package
reduce.random	See the general documentation under buildmer-package
calc.anova	See the general documentation under buildmer-package
calc.summary	See the general documentation under buildmer-package
ddf	The method used for calculating p -values if all smooth terms were eliminated and <code>calc.summary=TRUE</code> . Options are 'Wald' (default), 'Satterthwaite' (if package <code>lmerTest</code> is available), 'Kenward-Roger' (if packages <code>lmerTest</code> and <code>pbkrtest</code> are available), and 'lme4' (no p -values)
...	Additional options to be passed to <code>gamm4</code>

See Also

[buildmer-package](#)

Examples

```
library(buildmer)
m <- buildgamm4(f1 ~ s(timepoint,by=following) +
               s(participant,timepoint,by=following,bs='fs'),data=vowels)
```

buildglmmTMB

Use buildmer to perform stepwise elimination on glmmTMB models

Description

Use buildmer to perform stepwise elimination on glmmTMB models

Usage

```
buildglmmTMB(formula, data = NULL, family = gaussian(), cl = NULL,
             direction = c("order", "backward"), crit = "LRT", include = NULL,
             reduce.fixed = TRUE, reduce.random = TRUE, calc.summary = TRUE,
             ...)
```

Arguments

formula	See the general documentation under buildmer-package
data	See the general documentation under buildmer-package
family	See the general documentation under buildmer-package
cl	See the general documentation under buildmer-package
direction	See the general documentation under buildmer-package
crit	See the general documentation under buildmer-package
include	See the general documentation under buildmer-package
reduce.fixed	See the general documentation under buildmer-package
reduce.random	See the general documentation under buildmer-package
calc.summary	See the general documentation under buildmer-package
...	Additional options to be passed to glmmTMB

See Also

[buildmer-package](#)

Examples

```
library(buildmer)
m <- buildglmmTMB(Reaction ~ Days + (Days|Subject), data=lme4::sleepstudy)
```

buildgls	<i>Use buildmer to fit generalized-least-squares models using gls from nlme</i>
----------	---

Description

Use buildmer to fit generalized-least-squares models using gls from nlme

Usage

```
buildgls(formula, data = NULL, cl = NULL, direction = c("order",
  "backward"), crit = "LRT", include = NULL, calc.anova = TRUE,
  calc.summary = TRUE, ...)
```

Arguments

formula	See the general documentation under buildmer-package
data	See the general documentation under buildmer-package
cl	See the general documentation under buildmer-package
direction	See the general documentation under buildmer-package
crit	See the general documentation under buildmer-package
include	See the general documentation under buildmer-package
calc.anova	See the general documentation under buildmer-package
calc.summary	See the general documentation under buildmer-package
...	Additional options to be passed to <code>gls</code>

See Also

[buildmer-package](#)

Examples

```
library(buildmer)
library(nlme)
vowels$event <- with(vowels, interaction(participant, word))
m <- buildgls(f1 ~ timepoint*following, correlation=corAR1(form=~1|event), data=vowels)
```

buildjulia	<i>Use buildmer to perform stepwise elimination on models fit with Julia package MixedModels via JuliaCall</i>
------------	--

Description

Use buildmer to perform stepwise elimination on models fit with Julia package MixedModels via JuliaCall

Usage

```
buildjulia(formula, data = NULL, family = gaussian(), include = NULL,
  julia_family = gaussian(), julia_link = NULL, julia_fun = NULL,
  direction = c("order", "backward"), crit = "LRT",
  reduce.fixed = TRUE, reduce.random = TRUE, ...)
```

Arguments

formula	See the general documentation under buildmer-package
data	See the general documentation under buildmer-package
family	See the general documentation under buildmer-package
include	See the general documentation under buildmer-package
julia_family	For generalized linear mixed models, the name of the Julia function to evaluate to obtain the error distribution. Only used if family is non-Gaussian This should probably be the same as family but with an initial capital, with the notable exception of logistic regression: if the R family is binomial, the Julia family should be 'Bernoulli'
julia_link	For generalized linear mixed models, the name of the Julia function to evaluate to obtain the link function. Only used if family is non-Gaussian If not provided, Julia's default link for your error distribution is used
julia_fun	If you need to change some parameters in the Julia model object before Julia fit! is called, you can provide an R function to manipulate the unfitted Julia object here. This function should accept two arguments: the first is the julia structure, which is a list containing a call element you can use as a function to call Julia; the second argument is the R JuliaObject corresponding to the unfitted Julia model. This can be used to e.g. change optimizer parameters before the model is fitted
direction	See the general documentation under buildmer-package
crit	See the general documentation under buildmer-package
reduce.fixed	See the general documentation under buildmer-package
reduce.random	See the general documentation under buildmer-package
...	Additional options to be passed to <code>LinearMixedModel()</code> or <code>GeneralizedLinearMixedModel()</code>

See Also

[buildmer-package](#)

Examples

```
library(buildmer)
m <- buildjulia(f1 ~ vowel*timepoint*following + (1|participant) + (1|word),data=vowels)
```

buildlme	<i>Use buildmer to perform stepwise elimination of the fixed-effects part of mixed-effects models fit via lme from nlme</i>
----------	---

Description

Use buildmer to perform stepwise elimination of the fixed-effects part of mixed-effects models fit via lme from nlme

Usage

```
buildlme(formula, data = NULL, random, cl = NULL,
         direction = c("order", "backward"), crit = "LRT", include = NULL,
         calc.anova = TRUE, calc.summary = TRUE, ...)
```

Arguments

formula	See the general documentation under buildmer-package
data	See the general documentation under buildmer-package
random	The random-effects specification for the model. This is not manipulated by buildlme in any way!
cl	See the general documentation under buildmer-package
direction	See the general documentation under buildmer-package
crit	See the general documentation under buildmer-package
include	See the general documentation under buildmer-package
calc.anova	See the general documentation under buildmer-package
calc.summary	See the general documentation under buildmer-package
...	Additional options to be passed to lme

See Also

[buildmer-package](#)

Examples

```
library(buildmer)
m <- buildlme(Reaction ~ Days, data=lme4::sleepstudy, random=~Days|Subject)
```

buildmer	<i>Use buildmer to fit mixed-effects models using lmer/glmer from lme4</i>
----------	--

Description

Use buildmer to fit mixed-effects models using lmer/glmer from lme4

Usage

```
buildmer(formula, data = NULL, family = gaussian(), cl = NULL,
  direction = c("order", "backward"), crit = "LRT", include = NULL,
  reduce.fixed = TRUE, reduce.random = TRUE, calc.anova = TRUE,
  calc.summary = TRUE, ddf = "Wald", ...)
```

Arguments

formula	See the general documentation under buildmer-package
data	See the general documentation under buildmer-package
family	See the general documentation under buildmer-package
cl	See the general documentation under buildmer-package
direction	See the general documentation under buildmer-package
crit	See the general documentation under buildmer-package
include	See the general documentation under buildmer-package
reduce.fixed	See the general documentation under buildmer-package
reduce.random	See the general documentation under buildmer-package
calc.anova	See the general documentation under buildmer-package
calc.summary	See the general documentation under buildmer-package
ddf	The method used for calculating p -values if <code>calc.anova=TRUE</code> or <code>calc.summary=TRUE</code> . Options are 'Wald' (default), 'Satterthwaite' (if package <code>lmerTest</code> is available), 'Kenward-Roger' (if packages <code>lmerTest</code> and <code>pbkrtest</code> are available), and 'lme4' (no p -values)
...	Additional options to be passed to <code>lmer</code> , <code>glmer</code> , or <code>gamm4</code> . (They will also be passed to <code>(g)lm</code> in so far as they're applicable, so you can use arguments like <code>subset=...</code> and expect things to work. The single exception is the <code>control</code> argument, which is assumed to be meant only for <code>lme4</code> and not for <code>(g)lm</code> , and will <i>not</i> be passed on to <code>(g)lm</code> .)

Examples

```
library(buildmer)
m <- buildmer(Reaction ~ Days + (Days|Subject), lme4::sleepstudy)
```

buildmer-class	<i>The buildmer class</i>
----------------	---------------------------

Description

This is a simple convenience class that allows ‘anova()’ and ‘summary()’ calls to fall through to the underlying model object, while retaining buildmer’s iteration history. If you need to use the final model for other things, such as prediction, access it through the ‘model’ slot of the buildmer class object.

Slots

model The final model containing only the terms that survived elimination
 p Parameters used during the fitting process
 anova The model’s ANOVA, if the model was built with ‘anova=TRUE’
 summary The model’s summary, if the model was built with ‘summary=TRUE’

See Also

[buildmer()]

Examples

```
# Manually create a bare-bones buildmer object:
model <- lm(Sepal.Length ~ Petal.Length, iris)
p <- list(in.buildmer=FALSE)
library(buildmer)
bm <- mkBuildmer(model=model, p=p, anova=NULL, summary=NULL)
summary(bm)
```

buildmertree	<i>Use buildmer to perform stepwise elimination for the random-effects part of lmertree() and glmertree() models from package glmertree</i>
--------------	---

Description

Use buildmer to perform stepwise elimination for *the random-effects part* of lmertree() and glmertree() models from package glmertree

Usage

```
buildmertree(formula, data = NULL, family = gaussian(), cl = NULL,
  direction = "order", crit = "LL", include = NULL,
  calc.summary = TRUE, left = NULL, right = NULL, ...)
```


Arguments

formula	Either a glmertree formula, looking like <code>dep ~ left middle right</code> where the middle part is an lme4-style random-effects specification, or an ordinary formula (or buildmer term list thereof) specifying only the dependent variable and lme4-style random effects. In the latter case, the additional arguments <code>left</code> and <code>right</code> must be specified as one-sided formulas containing the fixed part of the model and the partitioning part, respectively
data	See the general documentation under buildmer-package
family	See the general documentation under buildmer-package
cl	See the general documentation under buildmer-package
direction	See the general documentation under buildmer-package
crit	See the general documentation under buildmer-package
include	See the general documentation under buildmer-package
calc.summary	See the general documentation under buildmer-package
left	The left part of the glmertree formula, used if formula does not contain glmertree-specific terms. Note that if <code>left</code> is specified when formula is in glmertree format, <code>left</code> overrides the formula specification!
right	The right part of the glmertree formula, used if formula does not contain glmertree-specific terms. Note that if <code>right</code> is specified when formula is in glmertree format, <code>right</code> overrides the formula specification!
...	Additional options to be passed to <code>lmtree</code> or <code>glmertree</code>

See Also

[buildmer-package](#)

Examples

```
library(buildmer)
m <- buildmertree(Reaction ~ 1 | (Days|Subject) | Days,crit='LL',direction='order',
                 data=lme4::sleepstudy,joint=FALSE)
m <- buildmertree(Reaction ~ 1 | (Days|Subject) | Days,crit='LL',direction='order',
                 data=lme4::sleepstudy,family=Gamma(link=identity),joint=FALSE)
```

buildmultinom	<i>Use buildmer to perform stepwise elimination for multinom models from package nnet</i>
---------------	---

Description

Use `buildmer` to perform stepwise elimination for multinom models from package `nnet`

Usage

```
buildmultinom(formula, data = NULL, cl = NULL, direction = c("order",
  "backward"), crit = "LRT", include = NULL, calc.summary = TRUE,
  ...)
```

Arguments

formula	See the general documentation under buildmer-package
data	See the general documentation under buildmer-package
cl	See the general documentation under buildmer-package
direction	See the general documentation under buildmer-package
crit	See the general documentation under buildmer-package
include	See the general documentation under buildmer-package
calc.summary	See the general documentation under buildmer-package
...	Additional options to be passed to multinom

See Also

[buildmer-package](#)

Examples

```
library(buildmer)
options(contrasts = c("contr.treatment", "contr.poly"))
library(MASS)
example(birthwt)
bwt.mu <- buildmultinom(low ~ age*lw*race*smoke,bwt)
```

conv

Test a model for convergence

Description

Test a model for convergence

Usage

```
conv(model, singular.ok = FALSE)
```

Arguments

model	The model object to test.
singular.ok	A logical indicating whether singular fits are accepted as ‘converged’ or not. Relevant only for lme4 models.

Value

Logical indicating whether the model converged.

Examples

```
library(buildmer)
library(lme4)
good1 <- lm(Reaction ~ Days,sleepstudy)
good2 <- lmer(Reaction ~ Days + (Days|Subject),sleepstudy)
bad <- lmer(Reaction ~ Days + (Days|Subject),sleepstudy,control=lmerControl(
  optimizer='bobyqa',optCtrl=list(maxfun=1)))
sapply(c(good1,good2,bad),conv)
```

diag, formula-method	<i>Diagonalize the random-effect covariance structure, possibly assisting convergence</i>
----------------------	---

Description

Diagonalize the random-effect covariance structure, possibly assisting convergence

Usage

```
## S4 method for signature 'formula'
diag(x)
```

Arguments

x A model formula.

Value

The formula with all random-effect correlations forced to zero, per Pinheiro & Bates (2000)

Examples

```
# 1. Create explicit columns for factor variables
library(buildmer)
vowels <- cbind(vowels,model.matrix(~vowel,vowels))
# 2. Create formula with diagonal covariance structure
form <- diag(f1 ~ (vowel1+vowel2+vowel3+vowel4)*timepoint*following +
  ((vowel1+vowel2+vowel3+vowel4)*timepoint*following | participant) +
  (timepoint | word))
# 3. Convert formula to buildmer terms list, grouping terms starting with 'vowel'
terms <- tabulate.formula(form,group='vowel[^:]')
# 4. Directly pass the terms object to buildmer(), using the hidden 'dep' argument to specify
# the dependent variable
m <- buildmer(terms,data=vowels,dep='f1')
```

migrant	<i>A very small data set from a pilot study on sound change.</i>
---------	--

Description

A very small data set from a pilot study on sound change.

Usage

```
data(migrant)
```

Format

A standard data frame.

remove.terms	<i>Remove terms from an lme4 formula</i>
--------------	--

Description

Remove terms from an lme4 formula

Usage

```
remove.terms(formula, remove)
```

Arguments

formula	The lme4 formula.
remove	A vector of terms to remove. To remove terms nested inside random-effect groups, use '(term group)' syntax. Note that marginality is respected, i.e. no effects will be removed if they participate in a higher-order interaction, and no fixed effects will be removed if a random slope is included over that fixed effect.

Examples

```
library(buildmer)
remove.terms(Reaction ~ Days + (Days|Subject), '(Days|Subject)')
# illustration of the marginality checking mechanism:
remove.terms(Reaction ~ Days + (Days|Subject), '(1|Subject)') #refuses to remove the term
remove.terms(Reaction ~ Days + (Days|Subject), c('(Days|Subject)', '(1|Subject)')) #also
#refuses to remove the term, because marginality is checked before removal!
step1 <- remove.terms(Reaction ~ Days + (Days|Subject), '(Days|Subject)')
step2 <- remove.terms(step1, '(1|Subject)') #works
```

tabulate.formula	<i>Parse a formula into a buildmer terms list</i>
------------------	---

Description

Parse a formula into a buildmer terms list

Usage

```
tabulate.formula(formula, group = NULL)
```

Arguments

formula	A formula.
group	A character vector of regular expressions. Terms matching the same regular expression are assigned the same block, and will be evaluated together in buildmer functions.

Value

A buildmer terms list, which is just a normal data frame.

See Also

buildmer-package

Examples

```
form <- diag(f1 ~ (vowel1+vowel2+vowel3+vowel4)*timepoint*following +
             ((vowel1+vowel2+vowel3+vowel4)*timepoint*following|participant) + (timepoint|word))
tabulate.formula(form)
tabulate.formula(form,group='vowel[1-4]')
```

vowels	<i>Vowel data from a pilot study.</i>
--------	---------------------------------------

Description

Vowel data from a pilot study.

Usage

```
data(vowels)
```

Format

A standard data frame.

Index

*Topic **datasets**

migrant, [20](#)

vowels, [21](#)

add.terms, [4](#)

build.formula, [5](#)

buildbam, [6](#)

buildcustom, [7](#)

buildgam, [2](#), [8](#)

buildgamm4, [9](#)

buildglmTMB, [10](#)

buildgls, [11](#)

buildjulia, [12](#)

buildlme, [14](#)

buildmer, [2](#), [15](#)

buildmer-class, [16](#)

buildmer-package, [2](#)

buildmertree, [2](#), [16](#)

buildmultinom, [17](#)

conv, [18](#)

diag, formula-method, [19](#)

migrant, [20](#)

mkBuildmer (buildmer-class), [16](#)

remove.terms, [20](#)

tabulate.formula, [2](#), [21](#)

vowels, [21](#)