Package ‘buildmer’

March 1, 2020

Title Stepwise Elimination and Term Reordering for Mixed-Effects Regression

Version 1.5

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, the Bayesian Information Criterion, or the explained deviance.

Depends R (>= 3.2)
Imports graphics, lme4, methods, mgev, nlme, plyr, stats, utils
Suggests GLMMadaptive, JuliaCall, MASS, gamm4, glmertree, glmmTMB, knitr, lmerTest, nnet, parallel, partykit, pbkrtest, rmarkdown

License FreeBSD
Encoding UTF-8
LazyData true
RoxygenNote 7.0.2

BugReports https://github.com/cvoeten/buildmer/issues
VignetteBuilder knitr

NeedsCompilation no

Author Cesko C. Voeten [aut, cre] (<https://orcid.org/0000-0003-4687-9973>)
Maintainer Cesko C. Voeten <cvoeten@gmail.com>

Repository CRAN
Date/Publication 2020-03-01 15:00:02 UTC

R topics documented:

buildmer-package ............................................................... 2
add.terms ............................................................................. 4
build.formula ....................................................................... 4
# 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 buildmer term list as obtained from `tabulate.formula`. In the latter formulation, you also need to specify a `dep='...'` argument specifying the dependent variable to go along with the term list. See `tabulate.formula` 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 makeCluster from package parallel to use for parallelizing the evaluation of terms. Note that, if and only if using the cl 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 c(‘order’, ‘backward’), 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 based on chi-square mixtures per Stram & Lee 1994 for random effects; this is the default), ‘LL’ (use the raw -2 log likelihood), ‘AIC’ (Akaike Information Criterion), ‘BIC’ (Bayesian Information Criterion), and ‘deviance’ (explained deviance – note that this is not a formal test).

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 formula argument. Useful for e.g. passing correlation structures in glmmTMB models.

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.
## add.terms

**Add terms to a formula**

### Description

Add terms to a formula

### Usage

```r
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

```r
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

**Convert a buildmer term list into a proper model formula**

### Description

Convert a buildmer term list into a proper model formula

### Usage

```r
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

```r
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

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

Description

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

Usage

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

Details

To work around an issue in bam(), you must make sure that your data do not contain a variable named 'intercept'.

lme4 random effects are supported: they will be automatically converted using re2mgcv.

As bam uses PQL, only crit='deviance' is supported for non-Gaussian errors.

See Also

buildmer-package

Examples

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

---

**buildcustom**  
Use buildmer to perform stepwise elimination using a custom fitting function

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(p, ref, alt) stop("'crit' not specified"), 
  include = NULL, 
  fit = function(p, formula) stop("'fit' not specified"), 
  elim = function(x) stop("'elim' not specified"), 
  REML = FALSE, 
  ... 
)

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
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)
REML  A logical indicating if the fitting function distinguishes between fits differing in fixed effects (for which p$reml will be set to FALSE) and fits differing only in the random part (for which p$reml will be TRUE).

...  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) {
  # Need to implement the function logic here
  # Example: build custom model based on formula
  # Function implementation goes here
# 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
if (inherits(test, 'try-error')) test else m

crit.F <- function (p,a,b) { # use whole-model F
  pvals <- anova(b)$'Pr(>F)'
  pvals[length(pvals)-1]
}
crit.Wilks <- function (p,a,b) {
  Lambda <- anova(b,test='Wilks')$Wilks[1]
p <- length(coef(b))
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
model <- 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
if (require('MASS')) model <- 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**

```r
buildgam(
  formula,
  data = NULL,
  family = gaussian(),
  quickstart = 0,
  cl = NULL,
  direction = c("order", "backward"),
  crit = "LRT",
  include = NULL,
)```
calc.anova = FALSE,
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
- **quickstart**: A numeric with values 0 to 5. If set to 1, will use bam to obtain starting values for gam's outer iteration, potentially resulting in a much faster fit for each model. If set to 2, will disregard ML/REML and always use bam's fREML. 3 also sets discrete=TRUE. Values between 3 and 4 fit the quickstart model to a subset of that value (e.g., quickstart=3.1 fits the quickstart model to 10% of the data, which is also the default if quickstart=3. Values between 4 and 5 do the same, but also set a very sloppy convergence tolerance of 0.2.
- **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 gam

Details

To work around an issue in gam(), you must make sure that your data do not contain a variable named 'intercept'.

lme4 random effects are supported: they will be automatically converted using re2mgcv.

If gam's optimizer argument is not set to use outer iteration, gam fits using PQL. In this scenario, only crit='deviance' is supported.

General families implemented in mgcv are supported, provided that they use normal formulas. Currently, this is only true of the cox.ph family. Because this family can only be fitted using REML, buildgam automatically sets gam's select argument to TRUE and prevents removal of parametric terms.

The quickstart function is experimental. If you desire more control (e.g., discrete=FALSE but use.chol=TRUE), additional options can be provided as extra arguments and will be passed on to bam as they are applicable. Note that quickstart needs to be larger than 0 to trigger the quickstart path at all.

If scaled-t errors are used (family=sca), the quickstart path will also provide initial values for the two theta parameters (corresponding to the degrees of freedom and the scale parameter), but only if your installation of package mgcv is at least at version 1.8-32.
See Also

buildmer-package

Examples

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

Usage

buildgamm(
  formula,
  data = NULL,
  family = gaussian(),
  cl = NULL,
  direction = c("order", "backward"),
  crit = "LRT",
  include = NULL,
  calc.anova = FALSE,
  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 gamm
Details

The fixed and random effects are to be passed as a single formula in lme4 format. This is internally split up into the appropriate fixed and random parts. Only a single grouping factor is allowed. The random-effect covariance matrix is always unstructured. If you want to use pdMat covariance structures, you must (a) not specify any lme4 random-effects term in the formula, and (b) specify your own custom random argument as part of the ... argument. Note that buildgamm will merely pass this through; no term reordering or stepwise elimination is done on a user-provided random argument.

See Also

buildmer-package

Examples

library(buildmer)
model <- buildgamm(f1 ~ s(timepoint,by=following) + (following|participant) +
                 s(participant,timepoint,by=following,bs='fs'),data=vowels)
buildGLMMadaptive

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`
- **ddf**: The method used for calculating p-values if all smooth terms were eliminated and `calc.anova=TRUE` or `calc.summary=TRUE`. Options are 'Wald' (default), 'Satterthwaite' (if package lmerTest is available), 'Kenward-Roger' (if packages lmerTest and pbkrtest are available), and 'lme4' (no p-values)
- **...**: Additional options to be passed to `gamm4`

Details

The fixed and random effects are to be passed as a single formula in `lme4 format`. This is internally split up into the appropriate fixed and random parts.

See Also

- `buildmer-package`

Examples

```r
library(buildmer)
if (requireNamespace('gamm4')) model <- buildgamm4(f1 ~ s(timepoint,by=following) +
  s(participant,timepoint,by=following,bs='fs'),data=vowels)
```

---

**Description**

Use `buildmer` to fit generalized linear mixed models using `mixed_model` from package GLMMadaptive
Usage

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

Arguments

formula A formula specifying both fixed and random effects using lme4 syntax. (Unlike mixed_model, buildGLMMadaptive does not use a separate random argument!)
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
...

Details

The fixed and random effects are to be passed as a single formula in lme4 format. This is internally split up into the appropriate fixed and random parts.

See Also

buildmer-package

Examples

# nonsensical model given these data
if (requireNamespace("GLMMadaptive")) model <- buildGLMMadaptive(stress ~ vowel + (vowel|word),
  family=binomial, data=vowels, nAGQ=1)
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,
  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.summary See the general documentation under buildmer-package
...
  Additional options to be passed to glmmTMB

See Also

buildmer-package

Examples

library(buildmer)
model <- if (requireNamespace('glmmTMB')) buildglmmTMB(Reaction ~ Days + (Days|Subject), data=lme4::sleepstudy)
buildgls

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

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 = FALSE,
  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 gls

Details

A workaround is included to prevent an error when the model matrix is of less than full rank. The summary output of such a model will look a bit strange!

See Also

buildmer-package
Examples

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

buildjulia

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

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",
  ...
)

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
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.

See the general documentation under `buildmer-package`.

`crit`  
See the general documentation under `buildmer-package`.

Additional options to be passed to `LinearMixedModel()` or `GeneralizedLinearMixedModel()`.

**Examples**

```r
if (requireNamespace('JuliaCall')) model <- buildjulia(f1 ~ vowel*timepoint*following + (1|participant) + (1|word), data=vowels)
```

`buildlme`  
*Use buildmer to perform stepwise elimination of mixed-effects models fit via lme from nlme*

**Description**

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

**Usage**

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

- **formula**: A formula specifying both fixed and random effects using lme4 syntax. (Unlike lme, buildlme does not use a separate random argument!)
- **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 lme

Details

The fixed and random effects are to be passed as a single formula in lme4 format. This is internally split up into the appropriate fixed and random parts. Only a single grouping factor is allowed. The random-effect covariance matrix is always unstructured. If you want to use pdMat covariance structures, you must (a) not specify any lme4 random-effects term in the formula, and (b) specify your own custom random argument as part of the ... argument. Note that buildlme will merely pass this through; no term reordering or stepwise elimination is done on a user-provided random argument.

See Also

- buildmer-package

Examples

```r
library(buildmer)
model <- buildlme(Reaction ~ Days + (Days|Subject), data=lme4::sleepstudy)
```

---

**buildmer**

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

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,
  calc.anova = FALSE,
  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

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 calc.anova=TRUE or calc.summary=TRUE. Options are 'Wald' (default), 'Satterthwaite' (if package lmerTest is available), 'Kenward-Roger' (if packages lmerTest and pbkrtest are available), and 'lme4' (no p-values)

...        Additional options to be passed to lmer, glmer, or gamm4. (They will also be passed to (g)lm in so far as they're applicable, so you can use arguments like subset=... and expect things to work. The single exception is the control argument, which is assumed to be meant only for lme4 and not for (g)lm, and will not be passed on to (g)lm)

Examples

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

#tests from github issue #2:
bm.test <- buildmer(cbind(incidence,size - incidence) ~ period + (! | herd),
  family=binomial,data=lme4::cbpp)
bm.test <- buildmer(cbind(incidence,size - incidence) ~ period + (! | herd),
  family=binomial,data=lme4::cbpp,direction='forward')
bm.test <- buildmer(cbind(incidence, size - incidence) ~ period + (1 | herd),
family=binomial, data=lme4::cbpp, crit='AIC')
bm.test <- buildmer(cbind(incidence, size - incidence) ~ period + (1 | herd),
family=binomial, data=lme4::cbpp, direction='forward', crit='AIC')

---

buildmer-class

The buildmer class

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

Use buildmer to perform stepwise elimination for lmertree() and glmertree() models from package glmertree

Description

Use buildmer to perform stepwise elimination for lmertree() and glmertree() models from package glmertree
Usage

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

Arguments

formula Either a glmertree formula, looking like dep ~ left | middle | right 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 the fixed and random effects for the regression part. In the latter case, the additional argument partitioning must be specified as a one-sided formula containing the partitioning part of the model.

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

... Additional options to be passed to lmerTree or glmertree. (They will also be passed to (g)lmTree in so far as they're applicable. The single exception is the control argument, which is assumed to be meant only for (g)lmTree and not for (g)lmTree, and will not be passed on to (g)lmTree)

Details

Note that the likelihood-ratio test is not available for glmertree models, as it cannot be assured that the models being compared are nested. The default is thus to use AIC. In the generalized case or when testing many partitioning variables, it is recommended to pass joint=FALSE, as this results in a dramatic speed gain and reduces the odds of the final glmer model failing to converge or converging singularly.

See Also

buildmer-package
Examples

```r
if (requireNamespace('glmertree')) {
  model <- buildmertree(Reaction ~ 1 | (Days|Subject) | Days, crit='LL', direction='order',
                        data=lme4::sleepstudy)

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

**buildmultinom**

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

Description

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

Usage

```r
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`
conv

Examples

if (requireNamespace('nnet') && require('MASS')) {
  options(contrasts = c("contr.treatment", "contr.poly"))
  example(birthwt)
  bwt.mu <- buildmultinom(low ~ age*lwt*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**  
Diagonalize the random-effect covariance structure, possibly assisting convergence

**Description**

Diagonalize the random-effect covariance structure, possibly assisting convergence

**Usage**

```r
## 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**

```r
# 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**  
A very small dataset from a pilot study on sound change.

**Description**

A very small dataset from a pilot study on sound change.

**Usage**

```r
data(migrant)
```
**re2mgcv**

**Format**

A standard data frame.

---

**re2mgcv**

*Convert lme4 random-effect terms to mgcv 're' smooths*

**Description**

Convert lme4 random-effect terms to mgcv 're' smooths

**Usage**

re2mgcv(formula, data)

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>formula</td>
<td>The lme4 formula.</td>
</tr>
<tr>
<td>data</td>
<td>The data.</td>
</tr>
</tbody>
</table>

**Examples**

```r
library(buildmer)
re <- re2mgcv(temp ~ angle + (1|replicate) + (1|recipe), lme4::cake)
model <- buildgam(re$formula, re$data, family = mgcv::scat)
# note: the below does NOT work, as the dependent variable is looked up in the data by name!
re <- re2mgcv(log(Reaction) ~ Days + (Days|Subject), lme4::sleepstudy)
```

---

**remove.terms**

*Remove terms from an lme4 formula*

**Description**

Remove terms from an lme4 formula

**Usage**

remove.terms(formula, remove)

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>formula</td>
<td>The lme4 formula.</td>
</tr>
<tr>
<td>remove</td>
<td>A vector of terms to remove. To remove terms nested inside random-effect groups, use `term</td>
</tr>
</tbody>
</table>
### Examples

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

---

### `tabulate.formula`

**Parse a formula into a buildmer terms list**

#### 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

```r
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

| vowels | Vowel data from a pilot study. |

**Description**

Vowel data from a pilot study.

**Usage**

```r
data(vowels)
```

**Format**

A standard data frame.
Index

*Topic **datasets**
  migrant, 24
  vowels, 27

add. terms, 4

build.formula, 4
buildbam, 5
buildcustom, 6
buildgam, 2, 8
buildgamm, 10
buildgamm4, 11
buildGLMMadaptive, 12
buildglmmTMB, 14
buildgls, 15
buildjulia, 16
buildlme, 17
buildmer, 2, 18
buildmer-class, 20
buildmer-package, 2
buildmertree, 2, 20
buildmultinom, 22

conv, 23

diag, formula-method, 24

migrant, 24
mkBuildmer (buildmer-class), 20

re2mgcv, 6, 9, 25
remove. terms, 25

tabulate, formula, 2, 26

vowels, 27