

# Package ‘buildmer’

December 17, 2017

**Title** Stepwise Elimination and Term Reordering for Mixed-Effects Regression

**Version** 0.1

**Description** The buildmer package attempts to build, from the user's specifications, the largest possible regression model that will still converge, and then eliminates all terms from it that do not significantly contribute to an improvement of model deviance. Optional reordering of the terms by their contribution to the deviance (like SPSS) is also supported.

**Depends** R (>= 3.2), mgcv, lme4

**Suggests** lmerTest, pbkrtest, gamm4

**License** 2-clause BSD

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 6.0.1

## R topics documented:

add.terms . . . . .	2
buildbam . . . . .	2
buildgam . . . . .	4
buildgamm . . . . .	5
buildgamm4 . . . . .	6
buildgls . . . . .	7
buildlme . . . . .	8
buildmer . . . . .	9
buildmer-class . . . . .	11
calcWald . . . . .	11
conv . . . . .	12
diag.formula-method . . . . .	12
has.smooth.terms . . . . .	13
hasREML . . . . .	13
is.random.term . . . . .	14
is.smooth.term . . . . .	14
mcmc2tex . . . . .	15

mer2tex . . . . . 15

remove.terms . . . . . 16

stepwise . . . . . 16

vowels . . . . . 17

**Index** 18

---

add.terms	<i>Add terms to a formula.</i>
-----------	--------------------------------

---

**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.

**See Also**

buildmer

---

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

---

**Description**

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

**Usage**

```
buildbam(formula, data, family = gaussian, reorder.terms = TRUE,
  cl = NULL, reduce.fixed = TRUE, reduce.random = TRUE,
  direction = "backward", crit = "LRT", calc.anova = TRUE,
  calc.summary = TRUE, ddf = "Wald", quiet = FALSE, ...)
```

**Arguments**

formula	The model formula for the maximal model you would like to fit, if possible. Supports lme4 random effects and gamm4 smooth terms.
data	The data to fit the models to.
family	The error distribution to use. Only relevant for generalized models; if the family is empty or 'gaussian', the models will be fit using lm(er), otherwise they will be fit using glm(er) with the specified error distribution passed through.
reorder.terms	Whether to reorder the terms by their contribution to the log-likelihood before testing them.
cl	An optional cluster object as returned by parallel::makeCluster() to use for parallelizing the evaluation of terms during the reordering step.
reduce.fixed	Whether to reduce the fixed-effect structure.
reduce.random	Whether to reduce the random-effect structure.
direction	The direction for stepwise elimination; either 'forward' or 'backward' (default). Both or neither are also understood.
crit	The criterion used to test terms for elimination. Possible options are 'LRT' (default), 'AIC', and 'BIC'.
calc.anova	Whether to also calculate the ANOVA table for the final model after term elimination.
calc.summary	Whether to also calculate the summary table for the final model after term elimination.
quiet	Whether to suppress progress messages.
...	Additional options to be passed to bam().

**Value**

A buildmer object containing the following slots:

- model: the final model containing only the terms that survived elimination
- p: the parameter list used in the various buildmer modules. Things of interest this list includes are, among others:
  - results: a dataframe containing the results of the elimination process
  - messages: any warning messages
- . This information is also printed as part of the show() method.
- summary: the model's summary, if 'calc.summary=TRUE' was passed
- anova: the model's anova, if 'calc.anova=TRUE' was passed

**See Also**

buildmer

buildgam

*Use buildmer to fit generalized additive models using gam() from mgcv***Description**

Use buildmer to fit generalized additive models using gam() from mgcv

**Usage**

```
buildgam(formula, data, family = gaussian, reorder.terms = TRUE,
  cl = NULL, reduce.fixed = TRUE, reduce.random = TRUE,
  direction = "backward", crit = "LRT", calc.anova = TRUE,
  calc.summary = TRUE, ddf = "Wald", quiet = FALSE, ...)
```

**Arguments**

formula	The model formula for the maximal model you would like to fit, if possible.
data	The data to fit the models to.
family	The error distribution to use. Only relevant for generalized models; if the family is empty or 'gaussian', the models will be fit using lm(er), otherwise they will be fit using glm(er) with the specified error distribution passed through.
reorder.terms	Whether to reorder the terms by their contribution to the log-likelihood before testing them.
cl	An optional cluster object as returned by parallel::makeCluster() to use for parallelizing the evaluation of terms during the reordering step.
reduce.fixed	Whether to reduce the fixed-effect structure.
reduce.random	Whether to reduce the random-effect structure.
direction	The direction for stepwise elimination; either 'forward' or 'backward' (default). Both or neither are also understood.
crit	The criterion used to test terms for elimination. Possible options are 'LRT' (default), 'AIC', and 'BIC'.
calc.anova	Whether to also calculate the ANOVA table for the final model after term elimination.
calc.summary	Whether to also calculate the summary table for the final model after term elimination.
quiet	Whether to suppress progress messages.
...	Additional options to be passed to gam().

**Value**

A buildmer object containing the following slots:

- `model`: the final model containing only the terms that survived elimination
- `p`: the parameter list used in the various buildmer modules. Things of interest this list includes are, among others:
  - `results`: a dataframe containing the results of the elimination process
  - `messages`: any warning messages
- This information is also printed as part of the `show()` method.
- `summary`: the model's summary, if `'calc.summary=TRUE'` was passed
- `anova`: the model's anova, if `'calc.anova=TRUE'` was passed

**See Also**

`buildmer`

---

`buildgamm`

*The logical extension of `buildgam()` to `buildgamm()` is not supported, because (i) `gamm` assumes you know what you're doing; (ii) the log-likelihood of a `gamm` object's `'lme'` item is not actually the log-likelihood of the final model; (iii) in my experience, `gamm` fits often fail to converge. If you are only using `gamm` for its 'true' random effects, use `buildgamm4()`. If you are using `gamm` for correlation structures, use `buildbam()` to fit AR(1) models. If you want more complex correlation structures, perform the stepwise elimination process by hand...*

---

**Description**

The logical extension of `buildgam()` to `buildgamm()` is not supported, because (i) `gamm` assumes you know what you're doing; (ii) the log-likelihood of a `gamm` object's `'lme'` item is not actually the log-likelihood of the final model; (iii) in my experience, `gamm` fits often fail to converge. If you are only using `gamm` for its 'true' random effects, use `buildgamm4()`. If you are using `gamm` for correlation structures, use `buildbam()` to fit AR(1) models. If you want more complex correlation structures, perform the stepwise elimination process by hand...

**Usage**

`buildgamm(...)`

**See Also**

`buildgamm4`, `buildbam`, `buildgam`

---

 buildgamm4

*Use buildmer to fit generalized additive models using gam() from mgcv*


---

## Description

Use buildmer to fit generalized additive models using gam() from mgcv

## Usage

```
buildgamm4(...)
```

## Arguments

...	Additional options to be passed to gam().
formula	The model formula for the maximal model you would like to fit, if possible. Supports lme4 random effects and gamm4 smooth terms.
data	The data to fit the models to.
family	The error distribution to use. Only relevant for generalized models; if the family is empty or 'gaussian', the models will be fit using lm(er), otherwise they will be fit using glm(er) with the specified error distribution passed through.
reorder.terms	Whether to reorder the terms by their contribution to the log-likelihood before testing them.
cl	An optional cluster object as returned by parallel::makeCluster() to use for parallelizing the evaluation of terms during the reordering step.
reduce.fixed	Whether to reduce the fixed-effect structure.
reduce.random	Whether to reduce the random-effect structure.
direction	The direction for stepwise elimination; either 'forward' or 'backward' (default). Both or neither are also understood.
crit	The criterion used to test terms for elimination. Possible options are 'LRT' (default), 'AIC', and 'BIC'.
calc.anova	Whether to also calculate the ANOVA table for the final model after term elimination.
calc.summary	Whether to also calculate the summary table for the final model after term elimination.
ddf	The method used for calculating p-values if all smooth terms were eliminated and summary=TRUE. Options are 'Wald' (default), 'Satterthwaite' (if lmerTest is available), 'Kenward-Roger' (if lmerTest and pbkrtest are available), and 'lme4' (no p-values).
quiet	Whether to suppress progress messages.

**Value**

A buildmer object containing the following slots:

- `model`: the final model containing only the terms that survived elimination
- `p`: the parameter list used in the various buildmer modules. Things of interest this list includes are, among others:
  - `results`: a dataframe containing the results of the elimination process
  - `messages`: any warning messages
- This information is also printed as part of the `show()` method.
- `summary`: the model's summary, if `'calc.summary=TRUE'` was passed
- `anova`: the model's anova, if `'calc.anova=TRUE'` was passed

**See Also**

`buildmer`

---

`buildgls`

*Use buildmer to perform stepwise elimination on gls models*

---

**Description**

Use buildmer to perform stepwise elimination on gls models

**Usage**

```
buildgls(formula, data, random, reorder.terms = TRUE, cl = NULL,
  reduce.fixed = TRUE, direction = "backward", crit = "LRT",
  calc.anova = TRUE, calc.summary = TRUE, quiet = FALSE, ...)
```

**Arguments**

<code>formula</code>	The model formula for the maximal model you would like to fit, if possible.
<code>data</code>	The data to fit the models to.
<code>reorder.terms</code>	Whether to reorder the terms by their contribution to the log-likelihood before testing them.
<code>cl</code>	An optional cluster object as returned by <code>parallel::makeCluster()</code> to use for parallelizing the evaluation of terms during the reordering step.
<code>reduce.fixed</code>	Whether to reduce the fixed-effect structure.
<code>direction</code>	The direction for stepwise elimination; either <code>'forward'</code> or <code>'backward'</code> (default). Both or neither are also understood.
<code>crit</code>	The criterion used to test terms for elimination. Possible options are <code>'LRT'</code> (default), <code>'AIC'</code> , and <code>'BIC'</code> .
<code>calc.anova</code>	Whether to also calculate the ANOVA table for the final model after term elimination.

<code>calc.summary</code>	Whether to also calculate the summary table for the final model after term elimination.
<code>quiet</code>	Whether to suppress progress messages.
<code>...</code>	Additional options to be passed to <code>lme()</code> .

**Value**

A buildmer object containing the following slots:

- `model`: the final model containing only the terms that survived elimination
- `p`: the parameter list used in the various buildmer modules. Things of interest this list includes are, among others:
  - `results`: a dataframe containing the results of the elimination process
  - `messages`: any warning messages
- This information is also printed as part of the `show()` method.
- `summary`: the model's summary, if `'calc.summary=TRUE'` was passed
- `anova`: the model's anova, if `'calc.anova=TRUE'` was passed

**See Also**

`buildmer`

---

<code>buildlme</code>	<i>Use buildmer to perform stepwise elimination on lme models</i>
-----------------------	---

---

**Description**

Use buildmer to perform stepwise elimination on lme models

**Usage**

```
buildlme(formula, data, random, reorder.terms = TRUE, cl = NULL,
  reduce.fixed = TRUE, direction = "backward", crit = "LRT",
  calc.anova = TRUE, calc.summary = TRUE, quiet = FALSE, ...)
```

**Arguments**

<code>formula</code>	The model formula for the maximal model you would like to fit, if possible.
<code>data</code>	The data to fit the models to.
<code>random</code>	The random-effects specification for the model. This is not manipulated by <code>buildlme()</code> in any way!
<code>reorder.terms</code>	Whether to reorder the terms by their contribution to the log-likelihood before testing them.
<code>cl</code>	An optional cluster object as returned by <code>parallel::makeCluster()</code> to use for parallelizing the evaluation of terms during the reordering step.



<code>reduce.fixed</code>	Whether to reduce the fixed-effect structure.
<code>direction</code>	The direction for stepwise elimination; either 'forward' or 'backward' (default). Both or neither are also understood.
<code>crit</code>	The criterion used to test terms for elimination. Possible options are 'LRT', 'AIC', and 'BIC'.
<code>calc.anova</code>	Whether to also calculate the ANOVA table for the final model after term elimination.
<code>calc.summary</code>	Whether to also calculate the summary table for the final model after term elimination.
<code>quiet</code>	Whether to suppress progress messages.
<code>...</code>	Additional options to be passed to <code>lme()</code> .

### Value

A `buildmer` object containing the following slots:

- `model`: the final model containing only the terms that survived elimination
- `p`: the parameter list used in the various `buildmer` modules. Things of interest this list includes are, among others:
  - `results`: a dataframe containing the results of the elimination process
  - `messages`: any warning messages
- . This information is also printed as part of the `show()` method.
- `summary`: the model's summary, if '`calc.summary=TRUE`' was passed
- `anova`: the model's anova, if '`calc.anova=TRUE`' was passed

### See Also

`buildmer`

---

<code>buildmer</code>	<i>Construct and fit as complete a model as possible, optionally reorder terms by their contribution to the log-likelihood, and perform stepwise elimination using the change in log-likelihood</i>
-----------------------	---

---

### Description

Construct and fit as complete a model as possible, optionally reorder terms by their contribution to the log-likelihood, and perform stepwise elimination using the change in log-likelihood

### Usage

```
buildmer(formula, data, family = gaussian, reorder.terms = TRUE,
  cl = NULL, reduce.fixed = TRUE, reduce.random = TRUE,
  direction = "backward", crit = "LRT", calc.anova = TRUE,
  calc.summary = TRUE, ddf = "Wald", quiet = FALSE, ...)
```

## Arguments

<code>formula</code>	The model formula for the maximal model you would like to fit, if possible. Supports lme4 random effects and gamm4 smooth terms.
<code>data</code>	The data to fit the models to.
<code>family</code>	The error distribution to use. Only relevant for generalized models; if the family is empty or 'gaussian', the models will be fit using lm(er), otherwise they will be fit using glm(er) with the specified error distribution passed through.
<code>reorder.terms</code>	Whether to reorder the terms by their contribution to the log-likelihood before testing them.
<code>cl</code>	An optional cluster object as returned by <code>parallel::makeCluster()</code> to use for parallelizing the evaluation of terms during the reordering step.
<code>reduce.fixed</code>	Whether to reduce the fixed-effect structure.
<code>reduce.random</code>	Whether to reduce the random-effect structure.
<code>direction</code>	The direction for stepwise elimination; either 'forward' or 'backward' (default). Both or neither are also understood.
<code>crit</code>	The criterion used to test terms for elimination. Possible options are 'LRT' (default), 'AIC', and 'BIC'.
<code>calc.anova</code>	Whether to also calculate the ANOVA table for the final model after term elimination. This is useful if you want to calculate degrees of freedom by Kenward-Roger approximation, in which case generating the ANOVA table (via <code>lmerTest</code> ) will be very slow, and preparing the ANOVA in advance can be advantageous.
<code>calc.summary</code>	Whether to also calculate the summary table for the final model after term elimination. This is useful if you want to calculate degrees of freedom by Kenward-Roger approximation (default), in which case generating the summary (via <code>lmerTest</code> ) will be very slow, and preparing the summary in advance can be advantageous.
<code>ddf</code>	The method used for calculating p-values if <code>summary=TRUE</code> . Options are 'Wald' (default), 'Satterthwaite' (if <code>lmerTest</code> is available), 'Kenward-Roger' (if <code>lmerTest</code> and <code>pbkrtest</code> are available), and 'lme4' (no p-values).
<code>quiet</code>	Whether to suppress progress messages.
<code>...</code>	Additional options to be passed to (g)lmer 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 (g)lmer and not for (g)lm, and will NOT be passed on to (g)lm.)

## Value

A `buildmer` object containing the following slots:

- `model`: the final model containing only the terms that survived elimination
- `p`: the parameter list used in the various `buildmer` modules. Things of interest this list includes are, among others:
  - `results`: a dataframe containing the results of the elimination process

- messages: any warning messages
- . This information is also printed as part of the show() method.
- summary: the model's summary, if 'calc.summary=TRUE' was passed
- anova: the model's anova, if 'calc.anova=TRUE' was passed

## Examples

```
buildmer(Reaction~Days+(Days|Subject), sleepstudy)
```

---

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

---

## Description

The buildmer class

## Arguments

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

---

calcWald	<i>Calculate p-values based on Wald z-scores</i>
----------	--

---

## Description

Calculate p-values based on Wald z-scores

## Usage

```
calcWald(table, i, sqrt = FALSE)
```

## Arguments

table	A coefficient table from a summary or anova output.
i	The number of the column in that table containing the t-values.
sqrt	Whether we're testing F values or t values (default).

## Value

The table augmented with p-values.

---

conv	<i>Test an mgcv or merMod (or equivalent) object for convergence</i>
------	--

---

**Description**

Test an mgcv or merMod (or equivalent) object for convergence

**Usage**

```
conv(model)
```

**Arguments**

model	The model object to test.
-------	---------------------------

**Value**

Whether the model converged or not.

---

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**

formula	A model formula.
---------	------------------

**Value**

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

---

has.smooth.terms	<i>Test whether a formula contains mgcv smooth terms</i>
------------------	--

---

**Description**

Test whether a formula contains mgcv smooth terms

**Usage**

```
has.smooth.terms(formula)
```

**Arguments**

formula	The formula.
---------	--------------

**Value**

A logical indicating whether the formula has any gamm4 terms.

---

hasREML	<i>Test whether a model was fit with REML</i>
---------	---

---

**Description**

Test whether a model was fit with REML

**Usage**

```
hasREML(model)
```

**Arguments**

model	A fitted model object.
-------	------------------------

**Value**

TRUE or FALSE if the model was a linear mixed-effects model that was fit with REML or not, respectively; NA otherwise.

---

is.random.term	<i>Test whether a formula term contains lme4 random terms</i>
----------------	---

---

**Description**

Test whether a formula term contains lme4 random terms

**Usage**

```
is.random.term(term)
```

**Arguments**

term	The term.
------	-----------

**Value**

A logical indicating whether the term was a random-effects term.

---

is.smooth.term	<i>Test whether a formula term is an mgcv smooth term</i>
----------------	---

---

**Description**

Test whether a formula term is an mgcv smooth term

**Usage**

```
is.smooth.term(term)
```

**Arguments**

term	The term.
------	-----------

**Value**

A logical indicating whether the term was a random-effects term.

---

mcmc2tex	<i>Convert an MCMCglmm model to LaTeX code (biased towards stress analysis)</i>
----------	---

---

**Description**

Convert an MCMCglmm model to LaTeX code (biased towards stress analysis)

**Usage**

```
mcmc2tex(model, aliases = list())
```

**Arguments**

model	The fitted model
aliases	A list of aliases translating summary terms to LaTeX code.

---

mer2tex	<i>Convert a buildmer (or compatible) model to LaTeX code</i>
---------	---

---

**Description**

Convert a buildmer (or compatible) model to LaTeX code

**Usage**

```
mer2tex(summary, aliases = list())
```

**Arguments**

summary	The model (or its summary) to convert.
aliases	A list of aliases translating summary terms to LaTeX code.

---

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

---

**Description**

Remove terms from an lme4 formula

**Usage**

```
remove.terms(formula, remove, formulize = T)
```

**Arguments**

<code>formula</code>	The lme4 formula.
<code>remove</code>	A vector of terms to remove. To remove terms nested inside random-effect groups, use ‘termgroup’ 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.
<code>formulize</code>	Whether to return a formula (default) or a simple list of terms.

**See Also**

`buildmer`

---

<code>stepwise</code>	<i>A simple interface to buildmer intended to mimic SPSS stepwise methods for term reordering and backward stepwise elimination</i>
-----------------------	---

---

**Description**

A simple interface to buildmer intended to mimic SPSS stepwise methods for term reordering and backward stepwise elimination

**Usage**

```
stepwise(formula, data, family = gaussian, ...)
```

**Arguments**

<code>formula</code>	The model formula for the maximal model you would like to fit, if possible. Supports lme4 random effects and gamm4 smooth terms.
<code>data</code>	The data to fit the models to.



family	The error distribution to use. Only relevant for generalized models; if the family is empty or 'gaussian', the models will be fit using lm(er), otherwise they will be fit using glm(er) with the specified error distribution passed through. Commonly-used options are either nothing/'gaussian' (linear regression), 'binomial' (logistic regression), or 'poisson' (loglin regression), although many other families exist (e.g. cloglog, ...).
...	Additional parameters that override buildmer defaults, see 'buildmer'.

**Value**

A buildmer object, which you can use summary() on to get a summary of the final model.

**Examples**

```
stepwise(Reaction~Days+(Days|Subject),sleepstudy)
```

---

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

---

**Description**

Vowel data from a pilot study.

**Usage**

```
data(vowels)
```

**Format**

A standard data frame.

**Examples**

```
#buildmer(f1 ~ vowel*timepoint*following + stress + information + (vowel*timepoint*following|participant),data=vowels)
buildmer(f1 ~ vowel + timepoint + stress + following + information + vowel:timepoint + timepoint:following + vowel:timepoint:following)
```

# Index

## \*Topic **datasets**

vowels, [17](#)

add.terms, [2](#)

buildbam, [2](#)

buildgam, [4](#)

buildgamm, [5](#)

buildgamm4, [6](#)

buildgls, [7](#)

buildlme, [8](#)

buildmer, [9](#)

buildmer-class, [11](#)

calcWald, [11](#)

conv, [12](#)

diag, formula-method, [12](#)

has.smooth.terms, [13](#)

hasREML, [13](#)

is.random.term, [14](#)

is.smooth.term, [14](#)

mcmc2tex, [15](#)

mer2tex, [15](#)

mkBuildmer (buildmer-class), [11](#)

remove.terms, [16](#)

stepwise, [16](#)

vowels, [17](#)