# Package ‘broom.mixed’

October 12, 2022

**Type**  Package  
**Title**  Tidying Methods for Mixed Models  
**Version**  0.2.9.4  
**Maintainer**  Ben Bolker <bolker@mcmaster.ca>  
**Description**  Convert fitted objects from various R mixed-model packages into tidy data frames along the lines of the 'broom' package. The package provides three S3 generics for each model: tidy(), which summarizes a model's statistical findings such as coefficients of a regression; augment(), which adds columns to the original data such as predictions, residuals and cluster assignments; and glance(), which provides a one-row summary of model-level statistics.  
**Imports**  broom, coda, dplyr, forcats, methods, nlme, purrr, stringr, tibble, tidyr, furrr  
**Suggests**  brms, dotwhisker, knitr, testthat, gamlss, gamlss.data, ggplot2, GLMMadaptive, glmmADMB, glmmTMB, ImerTest, lme4, Matrix, MCMCglmm, mgcv, pander, pbkrtest, rstan, rstanarm, rstantools, R2jags, TMB, rmarkdown  
**URL**  https://github.com/bbolker/broom.mixed  
**BugReports**  https://github.com/bbolker/broom.mixed/issues  
**License**  GPL-3  
**Encoding**  UTF-8  
**Additional_repositories**  http://bbolker.github.io/drat  
**VignetteBuilder**  knitr  
**RoxygenNote**  7.1.2  
**NeedsCompilation**  no  
**Author**  Ben Bolker [aut, cre] (<https://orcid.org/0000-0002-2127-0443>), David Robinson [aut], Dieter Menne [ctb], Jonah Gabry [ctb], Paul Buerkner [ctb],
augment.ranef.mer

Christopher Hua [ctb],
William Petry [ctb] (<https://orcid.org/0000-0002-5230-5987>),
Joshua Wiley [ctb] (<https://orcid.org/0000-0002-0271-6702>),
Patrick Kennedy [ctb],
Eduard Szöcs [ctb] (<https://orcid.org/0000-0001-5376-1194>, BASF SE),
Indrajeet Patil [ctb],
Vincent Arel-Bundock [ctb] (<https://orcid.org/0000-0003-2042-7063>),
Bill Denney [ctb],
Cory Brunson [ctb]

Repository CRAN
Date/Publication 2022-04-17 17:42:29 UTC

R topics documented:

- augment.ranef.mer .................................................. 2
- brms_tidiers ....................................................... 3
- compact ............................................................. 6
- fixef.MCMCglmm .................................................... 7
- gamlss_tidiers ...................................................... 8
- get_methods ........................................................ 9
- glmmadmb_tidiers ................................................... 9
- glmmTMB_tidiers .................................................... 12
- insert_NAs .......................................................... 15
- lme4_tidiers ........................................................ 15
- nlme_tidiers ......................................................... 19
- ranef.MCMCglmm .................................................... 22
- ranefLevels ......................................................... 23
- rstanarm_tidiers ................................................... 24
- stdranef ............................................................ 26
- tidy.MCMCglmm ..................................................... 27
- tidy.TMB ............................................................ 31
- tidy.varFunc ....................................................... 32
- unrowname .......................................................... 33

Index 34

------------------------------------------------------------------------
augment.ranef.mer Augmentation for random effects (for caterpillar plots etc.)
------------------------------------------------------------------------

Description

Augmentation for random effects (for caterpillar plots etc.)

Usage

```r
## S3 method for class 'ranef.mer'
augment(x, ci.level = 0.9, reorder = TRUE, order.var = 1, ...)
```
Arguments

- **x**: ranef (conditional mode) information from an lme4 fit, using `ranef(. , condVar=TRUE)
- **ci.level**: level for confidence intervals
- **reorder**: reorder levels by conditional mode values?
- **order.var**: numeric or character: which variable to use for ordering levels?
- **...**: additional arguments (unused: for generic consistency)

Examples

```r
if (require("lme4")) {
  load(system.file("extdata","lme4_example.rda",package="broom.mixed"))
  rr <- ranef(lmm1, condVar=TRUE)
  aa <- broom::augment(rr)

  ## Q-Q plot:
  if (require(ggplot2) && require(dplyr)) {
    g0 <- ggplot(aa,aes(estimate,qq,xmin=lb,xmax=ub))+
           geom_errorbarh(height=0)+
           geom_point()+facet_wrap(~variable, scale="free_x")
    g1 <- ggplot(aa,aes(estimate,level,xmin=lb,xmax=ub))+
           geom_errorbarh(height=0)+
           geom_vline(xintercept=0,lty=2)+
           geom_point()+facet_wrap(~variable, scale="free_x")
    aa2 <- group_by(aa, grp, level)
    aa3 <- mutate(aa2, keep=any(estimate/std.error>2))
    ggsplot(aa3, aes(estimate,level,xmin=lb,xmax=ub,colour=factor(keep)))+
           geom_errorbarh(height=0)+
           geom_vline(xintercept=0,lty=2)+
           geom_point()+facet_wrap(~variable, scale="free_x")+
           scale_colour_manual(values=c("black","red"), guide=FALSE)
  }
}
```

Description

These methods tidy the estimates from `brmsfit-objects` (fitted model objects from the `brms` package) into a summary.
Usage

## S3 method for class 'brmsfit'
tidy(x,
    parameters = NA,
    effects = c("fixed", "ran_pars"),
    robust = FALSE,
    conf.int = TRUE,
    conf.level = 0.95,
    conf.method = c("quantile", "HPDinterval"),
    fix.intercept = TRUE,
    exponentiate = FALSE,
    ...
)

## S3 method for class 'brmsfit'
glance(x, looic = FALSE, ...)

## S3 method for class 'brmsfit'
augment(x, data = stats::model.frame(x), newdata = NULL, se.fit = TRUE, ...)

Arguments

- **x** Fitted model object from the **brms** package. See **brmsfit-class**.
- **parameters** Names of parameters for which a summary should be returned, as given by a character vector or regular expressions. If NA (the default) summarized parameters are specified by the effects argument.
- **effects** A character vector including one or more of "fixed", "ran_vals", or "ran_pars". See the Value section for details.
- **robust** Whether to use median and median absolute deviation of the posterior distribution, rather than mean and standard deviation, to derive point estimates and uncertainty
- **conf.int** If TRUE columns for the lower (conf.low) and upper bounds (conf.high) of posterior uncertainty intervals are included.
- **conf.level** Defines the range of the posterior uncertainty conf.int, such that 100 * conf.level% of the parameter’s posterior distri- bution lies within the corresponding interval. Only used if conf.int = TRUE.
- **conf.method** method for computing confidence intervals ("quantile" or "HPDinterval")
- **fix.intercept** rename "Intercept" parameter to "(Intercept)", to match behaviour of other model types?
- **exponentiate** whether to exponentiate the fixed-effect coefficient estimates and confidence intervals (common for logistic regression); if TRUE, also scales the standard errors by the exponentiated coefficient, transforming them to the new scale
- **...** Extra arguments, not used
- **looic** Should the LOO Information Criterion (and related info) be included? See **loo.stanfit** for details. (This can be slow for models fit to large datasets.)
Value

All tidying methods return a data.frame without rownames. The structure depends on the method chosen.

When parameters = NA, the effects argument is used to determine which parameters to summarize.

Generally, tidy.brmsfit returns one row for each coefficient, with at least three columns:

- **term**: The name of the model parameter.
- **estimate**: A point estimate of the coefficient (mean or median).
- **std.error**: A standard error for the point estimate (sd or mad).

When effects = "fixed", only population-level effects are returned.

When effects = "ran_vals", only group-level effects are returned. In this case, two additional columns are added:

- **group**: The name of the grouping factor.
- **level**: The name of the level of the grouping factor.

Specifying effects = "ran_pars" selects the standard deviations and correlations of the group-level parameters.

If conf.int = TRUE, columns for the lower and upper bounds of the posterior conf.int computed.

Note

The names ‘fixed’, ‘ran_pars’, and ‘ran_vals’ (corresponding to "non-varying", "hierarchical", and "varying" respectively in previous versions of the package), while technically inappropriate in a Bayesian setting where "fixed" and "random" effects are not well-defined, are used for compatibility with other (frequentist) mixed model types.

At present, the components of parameter estimates are separated by parsing the column names of as_draws (e.g. r_patient[1,Intercept] for the random effect on the intercept for patient 1, or b_Trt1 for the fixed effect Trt1. We try to detect underscores in parameter names and warn, but detection may be imperfect.

See Also

brms,brmsfit-class
## compact

Remove NULL items in a vector or list

### Description

Remove NULL items in a vector or list

### Usage

```r
compact(x)
```

### Arguments

- `x`  
  a vector or list
**fixef.MCMCglmm**

Extract fixed effects from an `MCMCglmm` object

**Description**

Function designed to extract the fixed effects from an `MCMCglmm` model object. Can either extract all samples from the fixed effects posteriors or return the posterior means.

**Usage**

```r
## S3 method for class 'MCMCglmm'
fixef(object, use = c("all", "mean"), ...)
```

**Arguments**

- `object` An `MCMCglmm` model object to extract the effects from
- `use` A character string indicating whether to extract all posterior samples or the mean of the posteriors. Defaults to "all".
- `...` Arguments passed on to the worker function.

**Value**

A matrix of the fixed effects

**See Also**

- `ranef.MCMCglmm`

**Examples**

```r
## Not run:
# a simple MCMCglmm model
data(PlodiaPO)
m <- MCMCglmm(PO ~ 1, random = ~ FSfamily, data=PlodiaPO, verbose=FALSE)

# only extract average fixed effects
fixef(m, use = "mean")

# histogram of posterior samples of fixed effects
hist(fixef(m))
# matches the mean
rowMeans(fixef(m))
```

## End(Not run)
Tidying methods for "gamlss" objects from the gamlss package.

Usage

```r
## S3 method for class 'gamlss'
tidy(x, quick = FALSE, conf.int = FALSE, conf.level = 0.95, ...)
```

Arguments

- `x` A "gamlss" object
- `quick` Whether to perform a fast version, and return only the coefficients
- `conf.int` Whether to return confidence intervals
- `conf.level` Confidence level for CI
- `...` Arguments passed to `confint.gamlss`

Value

All tidying methods return a data.frame without rownames, whose structure depends on the method chosen.

A tibble with one row for each coefficient, containing columns:

- `parameter` Type of coefficient being estimated: mu, sigma, nu, or tau
- `term` Term in the model being estimated and tested
- `estimate` Estimated coefficient
- `std.error` Standard error
- `statistic` t-statistic
- `p.value` Two-sided p-value

Examples

```r
if (requireNamespace("gamlss", quietly = TRUE) &&
    requireNamespace("gamlss.data", quietly = TRUE)) {
  data(abdom, package="gamlss.data")
  ## Not run:
  mod <- gamlss(y~pb(x), sigma.fo=~pb(x), family=BCT,
                data=abdom, method=mixed(1,20))

  ## End(Not run)
  ## Load stored object
  mod <- readRDS(system.file("extdata", "gamlss_example.rds",
                        package="gamlss")
```
get_methods

```r
get_methods

package="broom.mixed")

tidy(mod)
```

---

**get_methods**

Retrieve all method/class combinations currently provided by the broom.mixed package

**Description**

Retrieve all method/class combinations currently provided by the broom.mixed package

**Usage**

```r
get_methods()
```

**Examples**

```r
print(get_methods(), n = Inf)
```

---

**glmmadmb_tidiers**

Tidying methods for glmmADMB models

**Description**

These methods tidy the coefficients of glmmADMB models

**Usage**

```r
## S3 method for class 'glmmadmb'
tidy(x,
effects = c("fixed", "ran_pars"),
component = "cond",
scales = NULL,
rans_prefix = NULL,
conf.int = FALSE,
conf.level = 0.95,
conf.method = "Wald",
...)

## S3 method for class 'glmmadmb'
augment(x, data = stats::model.frame(x), newdata, ...)

## S3 method for class 'glmmadmb'
glance(x, ...)
```
Arguments

x  An object of class `glmmadmb` `glmer`, or `nlmer` effects  A character vector including one or more of "fixed" (fixed-effect parameters), "ran_pars" (variances and covariances or standard deviations and correlations of random effect terms) or "ran_vals" (conditional modes/BLUPs/latent variable estimates) component  Which component(s) to report for (e.g., conditional, zero-inflation, dispersion: at present only works for "cond") scales  scales on which to report the variables: for random effects, the choices are "sd-cov" (standard deviations and correlations: the default if scales is NULL) or "varcov" (variances and covariances). NA means no transformation, appropriate e.g. for fixed effects; inverse-link transformations (exponentiation or logistic) are not yet implemented, but may be in the future. ran_prefix  a length-2 character vector specifying the strings to use as prefixes for self- (variance/standard deviation) and cross- (covariance/correlation) random effects terms conf.int  whether to include a confidence interval conf.level  confidence level for CI conf.method  method for computing confidence intervals (see `confint.merMod`) ...  extra arguments (not used) data  original data this was fitted on; if not given this will attempt to be reconstructed newdata  new data to be used for prediction; optional

Details

When the modeling was performed with `na.action = "na.omit"` (as is the typical default), rows with NA in the initial data are omitted entirely from the augmented data frame. When the modeling was performed with `na.action = "na.exclude"`, one should provide the original data as a second argument, at which point the augmented data will contain those rows (typically with NAs in place of the new columns). If the original data is not provided to `augment` and `na.action = "na.exclude"`, a warning is raised and the incomplete rows are dropped.

Value

All tidying methods return a `tbl_df` without rownames. The structure depends on the method chosen.
tidy returns one row for each estimated effect, either with groups depending on the effects parameter. It contains the columns
group  the group within which the random effect is being estimated: NA for fixed effects level  level within group (NA except for modes) term  term being estimated estimate  estimated coefficient std.error  standard error
### glmmadmb_tidiers

statistic  t- or $Z$-statistic (NA for modes)
p.value    P-value computed from t-statistic (may be missing/NA)

augment returns one row for each original observation, with columns (each prepended by a .) added. Included are the columns

- `.fitted` predicted values
- `.resid` residuals
- `.fixed` predicted values with no random effects

Also added for "merMod" objects, but not for "mer" objects, are values from the response object within the model (of type `lmResp`, `glmResp`, `nlsResp`, etc). These include `.mu`, `.offset`, `.sqrtXwt`, `.sqrtrwt`, `.eta`.

glance returns one row with the columns

- `sigma` the square root of the estimated residual variance
- `logLik` the data’s log-likelihood under the model
- `AIC` the Akaike Information Criterion
- `BIC` the Bayesian Information Criterion
- `deviance` deviance

### See Also

- `na.action`

### Examples

```r
if (require("glmmADMB") && require("lme4")) {
  ## original model
  ## Not run:
  data("sleepstudy", package="lme4")
  lmm1 <- glmmadmb(Reaction ~ Days + (Days | Subject), sleepstudy,
                   family="gaussian")

  ## End(Not run)
  ## load stored object
  load(system.file("extdata","glmmADMB_example.rda",package="broom.mixed"))
  tidy(lmm1, effects = "fixed")
  tidy(lmm1, effects = "fixed", conf.int=TRUE)
  ## tidy(lmm1, effects = "fixed", conf.int=TRUE, conf.method="profile")
  ## tidy(lmm1, effects = "ran_vals", conf.int=TRUE)
  head(augment(lmm1, sleepstudy))
  glance(lmm1)

  glmm1 <- glmmadmb(cbind(incidence, size - incidence) ~ period + (1 | herd),
                   data = cbpp, family = "binomial")
  tidy(glmm1)
  tidy(glmm1, effects = "fixed")
  head(augment(glmm1, cbpp))
```

---

```
---
```
Description

These methods tidy the coefficients of mixed effects models, particularly responses of the \texttt{merMod} class.

Usage

```r
## S3 method for class 'glmmTMB'
tidy(
  x,
  effects = c("ran_pars", "fixed"),
  component = c("cond", "zi"),
  scales = NULL,
  ran_prefix = NULL,
  conf.int = FALSE,
  conf.level = 0.95,
  conf.method = "Wald",
  exponentiate = FALSE,
  ...
)

## S3 method for class 'glmmTMB'
augment(x, data = stats::model.frame(x), newdata = NULL, ...)

## S3 method for class 'glmmTMB'
glance(x, ...)
```

Arguments

- **x**: An object of class \texttt{merMod}, such as those from \texttt{lmer}, \texttt{glmer}, or \texttt{nlmer}
- **effects**: A character vector including one or more of "fixed" (fixed-effect parameters), "ran_pars" (variances and covariances or standard deviations and correlations of random effect terms) or "ran_vals" (conditional modes/BLUPS/latent variable estimates)
- **component**: which component to extract (e.g. cond for conditional effects (i.e., traditional fixed effects); zi for zero-inflation model; disp for dispersion model)
- **scales**: scales on which to report the variables: for random effects, the choices are "sd-corr" (standard deviations and correlations: the default if scales is \texttt{NULL}) or "varcov" (variances and covariances). \texttt{NA} means no transformation, appropriate e.g. for fixed effects; inverse-link transformations (exponentiation or logistic) are not yet implemented, but may be in the future.
ran_prefix: a length-2 character vector specifying the strings to use as prefixes for self- (variance/standard deviation) and cross- (covariance/correlation) random effects terms

conf.int: whether to include a confidence interval

conf.level: confidence level for CI

conf.method: method for computing confidence intervals (see `confint.merMod`)

exponentiate: whether to exponentiate the fixed-effect coefficient estimates and confidence intervals (common for logistic regression); if TRUE, also scales the standard errors by the exponentiated coefficient, transforming them to the new scale

... extra arguments (not used)

data: original data this was fitted on; if not given this will attempt to be reconstructed

newdata: new data to be used for prediction; optional

Details

When the modeling was performed with `na.action = "na.omit"` (as is the typical default), rows with NA in the initial data are omitted entirely from the augmented data frame. When the modeling was performed with `na.action = "na.exclude"`, one should provide the original data as a second argument, at which point the augmented data will contain those rows (typically with NAs in place of the new columns). If the original data is not provided to `augment` and `na.action = "na.exclude"`, a warning is raised and the incomplete rows are dropped.

Value

All tidying methods return a tibble. The structure depends on the method chosen.

tidy returns one row for each estimated effect, either with groups depending on the effects parameter. It contains the columns

group: the group within which the random effect is being estimated: NA for fixed effects
level: level within group (NA except for modes)
term: term being estimated
estimate: estimated coefficient
std.error: standard error
statistic: t- or Z-statistic (NA for modes)
p.value: P-value computed from t-statistic (may be missing/NA)

augment returns one row for each original observation, with columns (each prepended by a .) added. Included are the columns

.fitted: predicted values
.resid: residuals
.fixed: predicted values with no random effects

glance returns one row with the columns

sigma: the square root of the estimated residual variance
logLik

the data’s log-likelihood under the model

AIC

the Akaike Information Criterion

BIC

the Bayesian Information Criterion

deviance

deviance

Note

zero-inflation parameters (including the intercept) are reported on the logit scale

See Also

na.action

Examples

if (require("glmmTMB") && require("lme4")
## &&
## make sure package versions are OK
## checkDepPackageVersion(dep_pkg = "TMB",
## this_pkg = "glmmTMB",
## warn = FALSE) &&
## checkDepPackageVersion(dep_pkg = "Matrix",
## this_pkg = "TMB",
## warn = FALSE))
{
  data("sleepstudy",package="lme4")
  ## original model:
  ## Not run:
  lmm1 <- glmmTMB(Reaction ~ Days + (Days | Subject), sleepstudy)
  ## End(Not run)
  ## load stored object
  L <- load(system.file("extdata","glmmTMB_example.rda",package="broom.mixed"))
  for (obj in L) {
    assign(obj, glmmTMB::up2date(get(obj)))
  }
  tidy(lmm1)
  tidy(lmm1, effects = "fixed")
  tidy(lmm1, effects = "fixed", conf.int=TRUE)
  tidy(lmm1, effects = "fixed", conf.int=TRUE, conf.method="uniroot")
  ## FIX: tidy(lmm1, effects = "ran_vals", conf.int=TRUE)
  head(augment(lmm1, sleepstudy))
  glance(lmm1)

  ## original model:
  ## glmm1 <- glmmTMB(incidence/size ~ period + (1 | herd),
  ## data = cbpp, family = binomial, weights=size)
  tidy(glmm1)
  tidy(glmm1, effects = "fixed")
  tidy(glmm1, effects = "fixed", exponentiate=TRUE)
```r
tidy(glmm1, effects = "fixed", conf.int=TRUE, exponentiate=TRUE)
head(augment(glmm1, cbpp))
head(augment(glmm1, cbpp, type.residuals="pearson"))
glance(glmm1)
## Not run:
## profile CIs - a little bit slower but more accurate
tidy(glmm1, effects = "fixed", conf.int=TRUE, exponentiate=TRUE, conf.method="profile")
## End(Not run)
```
conf.level = 0.95,
conf.method = "Wald",
ddf.method = NULL,
profile = NULL,
debug = FALSE,
...)

## S3 method for class 'rlmerMod'
tidy(
x,
effects = c("ran_pars", "fixed"),
scales = NULL,
exponentiate = FALSE,
ran_prefix = NULL,
conf.int = FALSE,
conf.level = NULL,
conf.method = "Wald",
ddf.method = NULL,
profile = NULL,
debug = FALSE,
...)

## S3 method for class 'merMod'
augment(x, data = stats::model.frame(x), newdata, ...)

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

Arguments

x An object of class merMod, such as those from lmer, glmer, or nlmer
effects A character vector including one or more of "fixed" (fixed-effect parameters); "ran_pars" (variances and covariances or standard deviations and correlations of random effect terms); "ran_vals" (conditional modes/BLUPs/latent variable estimates); or "ran_coefs" (predicted parameter values for each group, as returned by coef.merMod)
scales scales on which to report the variables: for random effects, the choices are "sd-cor" (standard deviations and correlations: the default if scales is NULL) or "vcov" (variances and covariances). NA means no transformation, appropriate e.g. for fixed effects.
exponentiate whether to exponentiate the fixed-effect coefficient estimates and confidence intervals (common for logistic regression); if TRUE, also scales the standard errors by the exponentiated coefficient, transforming them to the new scale
ran_prefix a length-2 character vector specifying the strings to use as prefixes for self- (variance/standard deviation) and cross- (covariance/correlation) random effects terms
When the modeling was performed with `na.action = "na.omit"` (as is the typical default), rows with NA in the initial data are omitted entirely from the augmented data frame. When the modeling was performed with `na.action = "na.exclude"`, one should provide the original data as a second argument, at which point the augmented data will contain those rows (typically with NAs in place of the new columns). If the original data is not provided to `augment` and `na.action = "na.exclude"`, a warning is raised and the incomplete rows are dropped.

Value

All tidying methods return a `data.frame` without rownames. The structure depends on the method chosen.

`tidy` returns one row for each estimated effect, either with groups depending on the `effects` parameter. It contains the columns

- `group`: the group within which the random effect is being estimated: "fixed" for fixed effects
- `level`: level within group (NA except for modes)
- `term`: term being estimated
- `estimate`: estimated coefficient
- `std.error`: standard error
- `statistic`: t- or Z-statistic (NA for modes)
- `p.value`: P-value computed from t-statistic (may be missing/NA)

`augment` returns one row for each original observation, with columns (each prepended by a .) added. Included are the columns

- `.fitted`: predicted values
- `.resid`: residuals
- `.fixed`: predicted values with no random effects
Also added for "merMod" objects, but not for "mer" objects, are values from the response object within the model (of type `lmResp`, `glmResp`, `nlsResp`, etc). These include ".mu", ".offset", ".sqrtXwt", ".sqrtrwt", ".eta".

glance returns one row with the columns

- `nobs`: the number of observations
- `sigma`: the square root of the estimated residual variance
- `logLik`: the data's log-likelihood under the model
- `AIC`: the Akaike Information Criterion
- `BIC`: the Bayesian Information Criterion
- `deviance`: deviance

See Also

- `na.action`

Examples

```r
if (require("lme4")) {
  ## original model
  ## Not run:
  lmm1 <- lmer(Reaction ~ Days + (Days | Subject), sleepstudy)

  ## End(Not run)

  ## load stored object
  load(system.file("extdata", "lme4_example.rda", package="broom.mixed"))
  (tt <- tidy(lmm1))
  tidy(lmm1, effects = "fixed")
  tidy(lmm1, effects = "fixed", conf.int=TRUE)
  tidy(lmm1, effects = "fixed", conf.int=TRUE, conf.method="profile")

  ## lmm1_prof <- profile(lmm1) # generated by extdata/runexamples
  tidy(lmm1, conf.int=TRUE, conf.method="profile", profile=lmm1_prof)

  ## conditional modes (group-level deviations from population-level estimate)
  tidy(lmm1, effects = "ran_vals", conf.int=TRUE)

  ## coefficients (group-level estimates)
  (rcoef1 <- tidy(lmm1, effects = "ran_coefs"))
  if (require(tidyr) && require(dplyr)) {
    ## reconstitute standard coefficient-by-level table
    spread(rcoef1,key=term,value=estimate)
    ## split ran_pars into type + term; sort fixed/sd/cor
    (tt %>% separate(term,c("type","term"),sep="--",fill="left")
      %>% arrange(!is.na(type),desc(type)))
  }
}
head(augment(lmm1, sleepstudy))
glance(lmm1)

glmm1 <- glmer(cbind(incidence, size - incidence) ~ period + (1 | herd),
                data = cbpp, family = binomial)
tidy(glmm1)
```
```r

tidy(glmm1, exponentiate=TRUE)
tidy(glmm1, effects = "fixed")
## suppress warning about influence.merMod
head(suppressWarnings(augment(glmm1, cbpp)))
glance(glmm1)

startvec <- c(Asym = 200, xmid = 725, scal = 350)
nm1 <- nlmer(circumference ~ SSlogis(age, Asym, xmid, scal) ~ Asym|Tree,
            Orange, start = startvec)
## suppress warnings about var-cov matrix ...
op <- options(warn=-1)
tidy(nm1)
tidy(nm1, effects = "fixed")
options(op)
head(augment(nm1, Orange))
glance(nm1)
detach("package:lme4")
}
if (require("lmerTest")) {
  lmm1 <- lmer(Reaction ~ Days + (Days | Subject), sleepstudy)
tidy(lmm1)
glance(lmm1)
detach("package:lmerTest") # clean up
}
```

---

**nlme_tidiers**

**Tidying methods for mixed effects models**

**Description**

These methods tidy the coefficients of mixed effects models of the `lme` class from functions of the `nlme` package.

**Usage**

```r

## S3 method for class 'lme'
tidy(
  x,
  effects = c("var_model", "ran_pars", "fixed"),
  scales = NULL,
  conf.int = FALSE,
  conf.level = 0.95,
  ...
)

## S3 method for class 'lme'
augment(x, data = x$data, newdata, ...)

## S3 method for class 'lme'
```

glance(x, ...)

## S3 method for class 'gls'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)

## S3 method for class 'gls'
augment(x, data = nlme::getData(x), newdata, ...)

Arguments

x  
An object of class lme, such as those from lme or nlme

effects  
One or more of "var_model", "ran_pars", "fixed", "ran_vals", and/or "ran_coefs".

scales  
scales on which to report the variables: for random effects, the choices are "sd-cor" (standard deviations and correlations: the default if scales is NULL) or "vcov" (variances and covariances). NA means no transformation, appropriate e.g. for fixed effects.

cnf.int  
whether to include a confidence interval

cnf.level  
confidence level for CI

...  
extra arguments (not used)

data  
original data this was fitted on; if not given this will attempt to be reconstructed

newdata  
ew data to be used for prediction; optional

Details

When the modeling was performed with na.action = "na.omit" (as is the typical default), rows with NA in the initial data are omitted entirely from the augmented data frame. When the modeling was performed with na.action = "na.exclude", one should provide the original data as a second argument, at which point the augmented data will contain those rows (typically with NAs in place of the new columns). If the original data is not provided to augment and na.action = "na.exclude", a warning is raised and the incomplete rows are dropped.

Value

All tidying methods return a data.frame without rownames. The structure depends on the method chosen.

tidy returns one row for each estimated effect, either random or fixed depending on the effects parameter. If effects = "ran_vals" (or "ran_pars"), it contains the columns

group  
the group within which the random effect is being estimated

level  
level within group

term  
term being estimated

estimated  
estimated coefficient

estimated  
This column is only included if some parameters are fixed. TRUE if the residual error is estimated and FALSE if the residual error is fixed.

If effects = "fixed", tidy returns the columns
### nlme_tidiers

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>term</td>
<td>fixed term being estimated</td>
</tr>
<tr>
<td>estimate</td>
<td>estimate of fixed effect</td>
</tr>
<tr>
<td>std.error</td>
<td>standard error</td>
</tr>
<tr>
<td>statistic</td>
<td>t-statistic</td>
</tr>
<tr>
<td>p.value</td>
<td>P-value computed from t-statistic</td>
</tr>
</tbody>
</table>

If `effects="var_model"` (the `weights` argument to the model), `tidy` returns the columns defined in the help for `tidy.varFunc`.

`augment` returns one row for each original observation, with columns (each prepended by a .) added. Included are the columns.

- `.fitted`  predicted values
- `.resid`  residuals
- `.fixed`  predicted values with no random effects

`glance` returns one row with the columns:

- `sigma`  the square root of the estimated residual variance
- `logLik`  the data's log-likelihood under the model
- `AIC`  the Akaike Information Criterion
- `BIC`  the Bayesian Information Criterion

#### See Also
- `na.action`

#### Examples

```r
if (require("nlme") && require("lme4")) {
  data("sleepstudy", package="lme4")
  ## original model
  ## Not run:
  lmm1 <- lme(Reaction ~ Days, random=~ Days|Subject, sleepstudy)
  ## End(Not run)

  ## load stored object
  load(system.file("extdata","nlme_example.rda", package="broom.mixed"))
  tidy(lmm1)
  tidy(lmm1, effects = "fixed")
  tidy(lmm1, conf.int = TRUE)
  tidy(lmm1, effects = "ran_pars")
  tidy(lmm1, effects = "ran_vals")
  tidy(lmm1, effects = "ran_coefs")
}
```
head(augment(lmm1, sleepstudy))
glance(lmm1)

startvec <- c(Asym = 200, xmid = 725, scal = 350)
nm1 <- nlme(circumference ~ SSlogis(age, Asym, xmid, scal),
            data = Orange,
            fixed = Asym + xmid + scal ~1,
            random = Asym ~1,
            start = startvec)
tidy(nm1)
tidy(nm1, effects = "fixed")
head(augment(nm1, Orange))
glance(nm1)

gls1 <- gls(follicles ~ sin(2*pi*Time) + cos(2*pi*Time), Ovary,
           correlation = corAR1(form = ~ 1 | Mare))
tidy(gls1)
glance(gls1)
head(augment(gls1))

---

### ranef.MCMCglmm

*Extract random effects from an MCMCglmm object*

**Description**

Function designed to extract the random effects from an MCMCglmm model object. Can either extract all samples from the random effects posteriors or return the posterior means.

**Usage**

```r
gls1 <- gls(follicles ~ sin(2*pi*Time) + cos(2*pi*Time), Ovary,
           correlation = corAR1(form = ~ 1 | Mare))
tidy(gls1)
glance(gls1)
head(augment(gls1))

```

**Arguments**

- `object` An MCMCglmm model object to extract the effects from
- `use` A character string indicating whether to extract all posterior samples or the mean of the posteriors. Defaults to "all".
- `...` Arguments passed on to the worker function.

**Value**

A matrix of the fixed effects

**See Also**

fixef.MCMCglmm
### Examples

```r
## Not run:
# a simple MCMCglmm model
data(PlodiaPO)
m <- MCMCglmm(PO ~ 1, random = ~ FSfamily, data = PlodiaPO, pr = TRUE, verbose = FALSE)

# only extract average fixed effects
head(ranef(m, use = "mean"))

# histogram of posterior samples of fixed effects
hist(ranef(m)[1, ])
# matches the mean
rowMeans(ranef(m)[1:6, ])

## End(Not run)
```

---

### Description

Extract the levels of factors used for random effects in `MCMCglmm` objects

### Usage

```r
ranefLevels(object, data, ...)  
```

### Arguments

- `object`: An `MCMCglmm` model object
- `data`: The dataset used for the model
- `...`: Not currently used

### See Also

`paramNamesMCMCglmm`, `ranef.MCMCglmm`

### Examples

```r
## Not run:
# a simple MCMCglmm model
data(PlodiaPO)
m <- MCMCglmm(PO ~ 1, random = ~ FSfamily, data = PlodiaPO, verbose = FALSE)

# extract the random effects levels
ranefLevels(m, PlodiaPO)

## End(Not run)
```
Description

These methods tidy the estimates from rstanarm fits (stan_glm, stan_glmer, etc.) into a summary.

Usage

```r
## S3 method for class 'stanreg'
tidy(x,
   effects = "fixed",
   conf.int = FALSE,
   conf.level = 0.9,
   conf.method = c("quantile", "HPDinterval"),
   ...
)
## S3 method for class 'stanreg'
glance(x, looic = FALSE, ...)
```

Arguments

- **x**: Fitted model object from the rstanarm package. See `stanreg-objects`.
- **effects**: A character vector including one or more of "fixed", "ran_vals", or "ran_pars". See the Value section for details.
- **conf.int**: If TRUE columns for the lower (conf.low) and upper (conf.high) bounds of the 100*prob% posterior uncertainty intervals are included. See `posterior_interval` for details.
- **conf.level**: See `posterior_interval`.
- **conf.method**: method for computing confidence intervals ("quantile" or "HPDinterval")
- **looic**: Should the LOO Information Criterion (and related info) be included? See `loo.stanfit` for details. (This can be slow for models fit to large datasets.)

Value

All tidying methods return a data.frame without rownames. The structure depends on the method chosen.

When effects="fixed" (the default), tidy.stanreg returns one row for each coefficient, with three columns:

- **term**: The name of the corresponding term in the model.
estimate  A point estimate of the coefficient (posterior median).
std.error  A standard error for the point estimate based on mad. See the Uncertainty estimates section in print.stanreg for more details.

For models with group-specific parameters (e.g., models fit with stan_glmer), setting effects="ran_vals" selects the group-level parameters instead of the non-varying regression coefficients. Additional columns are added indicating the level and group. Specifying effects="ran_pars" selects the standard deviations and (for certain models) correlations of the group-level parameters.

Setting effects="auxiliary" will select parameters other than those included by the other options. The particular parameters depend on which rstanarm modeling function was used to fit the model. For example, for models fit using stan_glm the overdispersion parameter is included if effects="aux", for stan_lm the auxiliary parameters include the residual SD, R^2, and log(fit_ratio), etc.

glance returns one row with the columns

algorithm  The algorithm used to fit the model.
pss  The posterior sample size (except for models fit using optimization).
nobs  The number of observations used to fit the model.
sigma  The square root of the estimated residual variance, if applicable. If not applicable (e.g., for binomial GLMs), sigma will be given the value 1 in the returned object.

If looic=TRUE, then the following additional columns are also included:

looic  The LOO Information Criterion.
elpd_loo  The expected log predictive density (elpd_loo = -2 * looic).
p_loo  The effective number of parameters.

See Also

summary,stanfit-method

Examples

```r
if (require("rstanarm")) {
## Not run:
#
'## original model
  fit <- stan_glmer(mpg ~ wt + (1|cyl) + (1+wt|gear), data = mtcars,
                   iter = 300, chains = 2)

## End(Not run)
## load example data
fit <- readRDS(system.file("extdata", "rstanarm_example.rds", package="broom.mixed"))

# non-varying ("population") parameters
tidy(fit, conf.int = TRUE, prob = 0.5)
tidy(fit, conf.int = TRUE, conf.method = "HPDinterval", prob = 0.5)
```
# hierarchical sd & correlation parameters
tidy(fit, effects = "ran_pars")

# group-specific deviations from "population" parameters
tidy(fit, effects = "ran_vals")

# glance method
glance(fit)
## Not run:
   glance(fit, looic = TRUE, cores = 1)
## End(Not run)
} ## if require("rstanarm")

---

stdranef  

**Extract standard deviation of "random" effects from an MCMCglmm object**

---

**Description**

Function designed to extract the standard deviation of the random effects from an MCMCglmm model object. Note that this is not the same as the posterior distribution of (co)variance matrices. It is based on the posterior distribution of the random effects. This also means it requires pr=TRUE to be set in the model for the information to be saved. Can optionally return standard deviation of random effects after back transforming to the response metric. Currently probabilities, but only for ordinal family models (family="ordinal").

**Usage**

`stdranef(object, which, type = c("lp", "response"), ...)`

**Arguments**

- **object**: An MCMCglmm model object to extract the effects from
- **which**: A list of random effects to extract or their numeric positions. If there are two numbers in a list, effects are simultaneous.
- **type**: A character string indicating whether to calculate the standard deviation on the linear predictor metric, ‘lp’ or response, ‘response’.
- **...**: Not currently used.

**Value**

A list of class postMCMCglmmRE with means (M) and individual estimates (Data)
## Not run:

```r
# a simple MCMCglmm model
data(PlodiaPO)
PlodiaPO <- within(PlodiaPO, {
  PO2 <- cut(PO, quantile(PO, c(0, .33, .66, 1)))
  plate <- factor(plate)
})

m <- MCMCglmm(PO2 ~ 1, random = ~ FSfamily + plate,
  family = "ordinal", data = PlodiaPO,
  prior = list(
    R = list(V = 1, fix = 1),
    G = list(
      G1 = list(V = 1, nu = .002),
      G2 = list(V = 1, nu = .002)
    )
  ), verbose=FALSE, thin=1, pr=TRUE)

# summary of the model
summary(m)

# examples of extracting standard deviations of
# different random effects on the linear predictor metric
# or after transformation to probabilities (only for ordinal)
stdranef(m, which = list(1), type = "lp")
stdranef(m, which = list(2), type = "lp")
stdranef(m, which = list(1, 2, c(1, 2)), type = "lp")
stdranef(m, type = "lp")

## error because no 3rd random effect
#stdranef(m, which = list(1, 2, 3), type = "lp")

stdranef(m, which = list("FSfamily", "plate"), type = "lp")

# mean standard deviations on the probability metric
# also the full distributions, if desired in the Data slot.
res <- stdranef(m, type = "response")
res$M # means
hist(res$Data$FSfamily[, 1]) # histogram

## End(Not run)
```

---

**tidy.MCMCglmm**

Tidying methods for MCMC (Stan, JAGS, etc.) fits

### Description

Tidying methods for MCMC (Stan, JAGS, etc.) fits
Usage

```r
## S3 method for class 'MCMCglmm'
tidy(x, effects = c("fixed", "ran_pars"), scales = NULL, ...)

tidyMCMC(
  x,
  pars,
  robust = FALSE,
  conf.int = FALSE,
  conf.level = 0.95,
  conf.method = c("quantile", "HPDinterval"),
  drop.pars = c("lp__", "deviance"),
  rhat = FALSE,
  ess = FALSE,
  index = FALSE,
  ...
)

## S3 method for class 'rjags'
tidy(
  x,
  robust = FALSE,
  conf.int = FALSE,
  conf.level = 0.95,
  conf.method = "quantile",
  ...
)

## S3 method for class 'stanfit'
tidy(
  x,
  pars,
  robust = FALSE,
  conf.int = FALSE,
  conf.level = 0.95,
  conf.method = c("quantile", "HPDinterval"),
  drop.pars = c("lp__", "deviance"),
  rhat = FALSE,
  ess = FALSE,
  index = FALSE,
  ...
)

## S3 method for class 'mcmc'
tidy(
  x,
  pars,
  robust = FALSE,
  ...}
```
tidy.MCMCglmm

conf.int = FALSE,
conf.level = 0.95,
conf.method = c("quantile", "HPDinterval"),
drop.pars = c("lp__", "deviance"),
rhat = FALSE,
ess = FALSE,
index = FALSE,
...
)

## S3 method for class 'mcmc.list'
tidy(
x, pars, robust = FALSE, conf.int = FALSE, conf.level = 0.95, conf.method = c("quantile", "HPDinterval"),
drop.pars = c("lp__", "deviance"), rhat = FALSE,
 ess = FALSE, index = FALSE, ...
)

Arguments

x a model fit to be converted to a data frame
effects which effects (fixed, random, etc.) to return
scales scales on which to report results
... mostly unused; for tidy.MCMCglmm, these represent options passed through to tidy.mcmc (e.g. robust, conf.int, conf.method, ...)
pars (character) specification of which parameters to include
robust use mean and standard deviation (if FALSE) or median and mean absolute deviation (if TRUE) to compute point estimates and uncertainty?
conf.int (logical) include confidence interval?
conf.level probability level for CI
conf.method method for computing confidence intervals ("quantile" or "HPDinterval")
drop.pars Parameters not to include in the output (such as log-probability information)
rhat, ess (logical) include Rhat and/or effective sample size estimates?
index Add index column, remove index from term. For example, term a[13] becomes term a and index 13.
Examples

```r
if (require("MCMCglmm")) {
    ## original model
    ## Not run:
    mm0 <- MCMCglmm(Reaction ~ Days,
                    random = ~Subject, data = sleepstudy,
                    nitt=4000,
                    pr = TRUE
    )

    ## End(Not run)
    ## load stored object
    load(system.file("extdata","MCMCglmm_example.rda",
                     package="broom.mixed"))
    tidy(mm0)
    tidy(mm1)
    tidy(mm2)
    tail(tidy(mm0, effects="ran_vals"))
}
```

# Using example from "RStan Getting Started"
# https://github.com/stan-dev/rstan/wiki/RStan-Getting-Started

```r
model_file <- system.file("extdata","8schools.stan", package = "broom.mixed")
schools_dat <- list(J = 8,
                     y = c(28, 8, -3, 7, -1, 1, 18, 12),
                     sigma = c(15, 10, 16, 11, 9, 11, 10, 18))

## original model
## Not run:
set.seed(2015)
rstan_example <- rstan::stan(file = model_file, data = schools_dat,
                             iter = 1000, chains = 2, save_dso = FALSE)

## End(Not run)
if (require(rstan)) {
    ## load stored object
    rstan_example <- readRDS(system.file("extdata","rstan_example.rds", package = "broom.mixed"))
    tidy(rstan_example)
    tidy(rstan_example, conf.int = TRUE, pars = "theta")
    td_mean <- tidy(rstan_example, conf.int = TRUE)
    td_median <- tidy(rstan_example, conf.int = TRUE, robust = TRUE)

    if (require(dplyr) & & require(ggplot2)) {
        tds <- (dplyr::bind_rows(list(mean=td_mean, median=td_median), .id="method")
               %>% mutate(type=ifelse(grepl("^theta",term),"theta",
                                 ifelse(grepl("^eta",term),"eta","other")))
        )

        ggplot(tds, aes(estimate, term)) +
        geom_errorbarh(aes(xmin = conf.low, xmax = conf.high, height=0)) +
        geom_point(aes(color = method))
    }
}
```
Tidy.TMB  

### Description

Tidying methods for TMB models

### Usage

```r
## S3 method for class 'TMB'
tidy(x,  
effects = c("fixed", "random"),  
conf.int = FALSE,  
conf.level = 0.95,  
conf.method = c("wald", "uniroot", "profile"),  
...)
```

### Arguments

- **x**: An object of class TMB (you may need to use `class(obj) <- "TMB"` on your results from TMB)
- **effects**: which effects should be returned?
- **conf.int**: whether to include a confidence interval
- **conf.level**: confidence level for CI
- **conf.method**: method for computing confidence intervals
- **...**: additional arguments passed to `confint` function (tmbroot, tmbprofile)
tidy.varFunc

Examples

if (require("TMB")) {
  ## Not run:
  runExample("simple", thisR=TRUE)
  class(obj) <- "TMB"
  tidy(obj, conf.int=TRUE, conf.method="wald")

  ## End(Not run)
  ## Not run: tidy(obj, conf.int=TRUE, conf.method="univar")
  ## Not run: tidy(obj, conf.int=TRUE, conf.method="profile")
}

---

**tidy.varFunc**

*Tidy variance structure for the nlme package.*

**Description**

Returns a tibble with the following columns:

- `grouptype` of varFunc, along with the right hand side of the formula in parentheses e.g. "varExp(age | Sex)".
- `term` terms included in the formula of the variance model, specifically the names of the coefficients. If the value is fixed, it will be appended with " ; fixed".
- `estimated` estimated coefficient
- `estimated` This column is only included if some parameters are fixed. TRUE if the parameter is estimated and FALSE if the parameter is fixed.

**Usage**

```r
## S3 method for class 'varFunc'
tidy(x, ...)
## S3 method for class 'varComb'
tidy(x, ...)
```

**Arguments**

- `x` An object of class varFunc, such as those used as the weights argument from the nlme package
- `...` Ignored

**Value**

If the varFunc is uninitialized or has no parameters, the function will return an empty tibble. Otherwise, it will return a tibble with names described in the details section.
Examples

## Not run:

```r
if (require("nlme")) {
  ChickWeight_arbitrary_group <- datasets::ChickWeight
  ChickWeight_arbitrary_group$group_arb_n <-
    1 + (as.integer(ChickWeight_arbitrary_group$Chick) >
      median(as.integer(ChickWeight_arbitrary_group$Chick))
  )
  ChickWeight_arbitrary_group$group_arb <- c("low", "high")[ChickWeight_arbitrary_group$group_arb_n]

  fit_with_fixed <-
    lme(
      weight ~ Diet * Time,
      random = ~Time | Chick,
      data = ChickWeight_arbitrary_group,
      weights=varIdent(fixed=c("low"=5), form=-1|group_arb)
    )
  # Show all parameters
  tidy(fit_with_fixed)
  # Exclude fixed parameters
  tidy(fit_with_fixed) %>%
    filter(across(any_of("estimated"), ~.x))
}
## End(Not run)
```

---

**unrowname**

*strip rownames from an object*

**Description**

strip rownames from an object

**Usage**

`unrowname(x)`

**Arguments**

- `x` a data frame
Index

augment.brmsfit(brms tragediers), 3
augment.glmmadmb(glmmadmb tragediers), 9
augment.glmmTMB(glmmTMB tragediers), 12
augment.gls(nlme trigiers), 19
augment.lme(nlme tragediers), 19
augment.merMod(lme4 tragediers), 15
augment.ranef.mer, 2
brms, 5
brms tragediers, 3
coeff.merMod, 16
compact, 6
confint.merMod, 10, 13
fixef.MCMCglmm, 7, 22
gamlss tragediers, 8
get_methods, 9
glance.allFit(lme4 tragediers), 15
glance.brmsfit(brms tragediers), 3
glance.glmmadmb(glmmadmb tragediers), 9
glance.glmmTMB(glmmTMB tragediers), 12
glance.lme(nlme tragediers), 19
glance.merMod(lme4 tragediers), 15

insert_NAs, 15
lme4 tragediers, 15
loo.stanfit, 4, 24
mad, 25
mcmc tragediers(tidy.MCMCglmm), 27
na.action, 11, 14, 18, 21
nlme tragediers, 19

paramNamesMCMCglmm, 23

posterior_interval, 24
print.stanreg, 25
ranef.MCMCglmm, 7, 22, 23
raneflevels, 23
rstanarm tragediers, 24
stan_glm, 25
stan_glmer, 25
stan_lm, 25
stdranef, 26
summary.lmerModLmerTest, 17
tidy.allFit(lme4 tragediers), 15
tidy.brmsfit(brms tragediers), 3
tidy.gamlass(gamlss tragediers), 8
tidy.glmmadmb(glmmadmb tragediers), 9
tidy.glmmTMB(glmmTMB tragediers), 12
tidy.gls(nlme tragediers), 19
tidy.lme(nlme tragediers), 19
tidy.mcmc(tidy.MCMCglmm), 27
tidy.MCMCglmm, 27
tidy.merMod(lme4 tragediers), 15
tidy.rjags(tidy.MCMCglmm), 27
tidy.rlmerMod(lme4 tragediers), 15
tidy.stanfit(tidy.MCMCglmm), 27
tidy.stanreg(rstanarm tragediers), 24
tidy.TMB, 31
tidy.varComb(tidy.varFunc), 32
tidy.varFunc, 32
tidyMCMC(tidy.MCMCglmm), 27

unrowname, 33