Package ‘broom.mixed’

July 7, 2021

Type Package

Title Tidying Methods for Mixed Models

Version 0.2.7

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, methods, nlme, purrr, stringr, tibble, tidy

Suggests brms, dotwhisker, knitr, testthat, gamlss, gamlss.data, ggplot2, GLMMadaptive, glmmADMB, glmmTMB, ImerTest, lme4, Matrix, MCMCglmm, mgcv, pander, pbkrtest, rstan, rstanarm, R2jags, TMB, rmarkdown

URL https://github.com/bbolker/broom.mixed

BugReports https://github.com/bbolker/broom.mixed/issues

License GPL-3

RoxygenNote 7.1.1

Encoding UTF-8

Additional_repositories http://bbolker.github.io/drat

VignetteBuilder knitr

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>)

Repository CRAN
Date/Publication 2021-07-07 15:20:02 UTC

R topics documented:

- augment.ranef.mer
- brms_tidiers
- compact
- fixef.MCMCglmm
- gamlss_tidiers
- glmmadmb_tidiers
- glmmTMB_tidiers
- insert_NAs
- lme4_tidiers
- nlme_tidiers
- ranef.MCMCglmm
- ranefLevels
- rstanarm_tidiers
- stdranef
- tidy.MCMCglmm
- tidy.TMB
- unrowname

Index 32

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

- \textit{x} \quad \text{ranef (conditional mode) information from an lme4 fit, using \texttt{ranef(., condVar=TRUE)}}
- \textit{ci.level} \quad \text{level for confidence intervals}
- \textit{reorder} \quad \text{reorder levels by conditional mode values?}
- \textit{order.var} \quad \text{numeric or character: which variable to use for ordering levels?}
- ... \quad \text{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")
    ## regular caterpillar plot:
    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")
    ## emphasize extreme values
    aa2 <- group_by(aa,grp,level)
    aa3 <- mutate(aa2, keep=any(estimate/std.error>2))
    ## Update caterpillar plot with extreme levels highlighted
    ## (highlight all groups with *either* extreme intercept *or* extreme slope)
    ggplot(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 \texttt{brmsfit-objects} (fitted model objects from the \texttt{brms} package) into a summary.
Usage

```r
## 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,
  ...
)
```

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

```r
## 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 $100 \times \text{conf.level}\%$ of the parameter’s posterior distribution 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?
- **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.)
- **data**: data frame
- **newdata**: new data frame
- **se.fit**: return standard errors of fit?
### 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 `posterior_samples` (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`

### Examples

```r
## original model
## Not run:
brms_crossedRE <- brm(mpg ~ wt + (1|cyl) + (1+wt|gear), data = mtcars,
                       iter = 500, chains = 2)
## End(Not run)

if (.Platform$OS.type != "windows" && require("brms")) {
  ## too slow on Windows, skip (>5 seconds on r-devel-windows)
  ## load stored object
  brms_crossedRE <- brm(mpg ~ wt + (1|cyl) + (1+wt|gear), data = mtcars,
                        iter = 500, chains = 2)
```
load(system.file("extdata", "brms_example.rda", package="broom.mixed"))

fit <- brms_crossedRE
tidy(fit)
tidy(fit, parameters = "sd", conf.int = FALSE)
tidy(fit, effects = "fixed", conf.method="HPDinterval")
tidy(fit, effects = "ran_vals")
tidy(fit, effects = "ran_pars", robust = TRUE)
# glance method
glance(fit)
## this example will give a warning that it should be run with
## reloo=TRUE; however, doing this will fail
## because the \code{fit} object has been stripped down to save space
suppressWarnings(glance(fit, looic = TRUE, cores = 1))
head(augment(fit))
}

---

compact Remove NULL items in a vector or list

Description

Remove NULL items in a vector or list

Usage

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

## 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)
```
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="broom.mixed"))
  tidy(mod)
}
```

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,
  ran_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
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", ".sqrtrwt", ".sqrtXwt", ".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))
  glance(glmm1)
}
```

---

**glmmTMB_tidiers**

**Tidying methods for glmmTMB models**

**Description**

These methods tidy the coefficients of mixed effects models, particularly responses of the `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 `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) 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-cor" (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`)
- **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
- **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
- **...**: extra arguments (not used)

### 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

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

  ## End(Not run)
  ## load stored object
  load(system.file("extdata","glmmTMB_example.rda",package="broom.mixed"))
```
insert_NAs

insert_NAs(data frame that has one row for each non-NA row in original)

Usage

insert_NAs(x, original)

Arguments

x data frame that has one row for each non-NA row in original
original data frame with NAs
lme4_tidiers

Tidying methods for mixed effects models

Description

These methods tidy the coefficients of mixed effects models, particularly responses of the merMod class.

Usage

```
## S3 method for class 'merMod'
tidy(
  x,
  effects = c("ran_pars", "fixed"),
  scales = NULL,
  exponentiate = FALSE,
  ran_prefix = NULL,
  conf.int = FALSE,
  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 = 0.95,
  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, ...)
```
lme4_tidiers

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
- **conf.int**: whether to include a confidence interval
- **conf.level**: confidence level for CI
- **conf.method**: method for computing confidence intervals (see `lme4::confint.merMod`)
- **ddf.method**: the method for computing the degrees of freedom and t-statistics (only applicable when using the `lmerTest` package: see `summary.lmerModLmerTest`)
- **profile**: pre-computed profile object, for speed when using `conf.method="profile"`
- **debug**: print debugging output?
- **...**: Additional arguments (passed to `confint.merMod` for `tidy`; `augment_columns` for `augment`; ignored for `glance`)
- **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 `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
.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

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)
nlme_tidiers

Tidying methods for mixed effects models

Description

These methods tidy the coefficients of mixed effects models of the \texttt{lme} class from functions of the \texttt{nlme} package.
Usage

```r
## S3 method for class 'lme'
tidy(
  x,
  effects = c("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 "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.
- **conf.int**: whether to include a confidence interval
- **conf.level**: confidence level for CI
- **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 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
estimate estimated coefficient

If effects="fixed", tidy returns the columns

term fixed term being estimated
estimate estimate of fixed effect
std.error standard error
statistic t-statistic
p.value P-value computed from t-statistic

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))
}
```

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

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

Description

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

Usage

ranefLevels(object, data, ...)

Arguments

  object  An MCMCglmm model object
  data    The dataset used for the model
           Not currently used
See Also

\texttt{paramNamesMCMCglmm}, \texttt{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)
```

---

**rstanarm_tidiers**

**Tidying methods for an rstanarm model**

**Description**

These methods tidy the estimates from \texttt{stanreg-objects} (fitted model objects from the \texttt{rstanarm} package) 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 \texttt{rstanarm} package. See \texttt{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 (\texttt{conf.low}) and upper (\texttt{conf.high}) bounds of the 100\*\texttt{prob} posterior uncertainty intervals are included. See \texttt{posterior_interval.stanreg} for details.
- **conf.level**: See \texttt{posterior_interval.stanreg}.  

```
conf.method  method for computing confidence intervals ("quantile" or "HPDinterval")

... For glance, if looic=TRUE, optional arguments to loo.stanreg.

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:

<table>
<thead>
<tr>
<th>term</th>
<th>The name of the corresponding term in the model.</th>
</tr>
</thead>
<tbody>
<tr>
<td>estimate</td>
<td>A point estimate of the coefficient (posterior median).</td>
</tr>
<tr>
<td>std.error</td>
<td>A standard error for the point estimate based on mad. See the Uncertainty estimates section in print.stanreg for more details.</td>
</tr>
</tbody>
</table>

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
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 simulataneous.

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)

Examples

```r
## Not run:
# 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")
```
tidy.MCMCglmm

```r
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,
...)
```

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

```r
## 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'

```r
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.list'

```r
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
tidy.MCMCglmm

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
  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)) +
  facet_wrap(~type,scale="free",ncol=1)
} ## require(dplyr,ggplot2)

if (require(R2jags)) {
  ## see help("jags",package="R2jags")
  ## and example("jags",package="R2jags")
  ## for details
  ## load stored object
  R2jags_example <- readRDS(system.file("extdata", "R2jags_example.rds", package = "broom.mixed"))
  tidy(R2jags_example)
  tidy(R2jags_example, conf.int=TRUE, conf.method="quantile")
}

---

**tidy.TMB**  
Tidying methods for TMB models

**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)
unrowname

which effects should be returned?
whether to include a confidence interval
certainty level for CI
method for computing confidence intervals
additional arguments passed to confint function (tmbroot, tmbprofile)

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

unrowname

strip rownames from an object

Description

strip rownames from an object

Usage

unrowname(x)

Arguments

x a data frame
augment.brmsfit(brms_tidiers), 3
augment.glmmadmb(glmmadmb_tidiers), 8
augment.glmmTMB(glmmTMB_tidiers), 11
augment.gls(nlme_tidiers), 18
augment.lme(nlme_tidiers), 18
augment.merMod(lme4_tidiers), 15
augment.ranef.mer, 2
brms, 5
brms_tidiers, 3
coeff.merMod, 16
compact, 6
confint.merMod, 9, 12
fixef.MCMCglmm, 6, 22
gamlss_tidiers, 7
glance.brmsfit(brms_tidiers), 3
glance.glmmadmb(glmmadmb_tidiers), 8
glance.glmmTMB(glmmTMB_tidiers), 11
glance.lme(nlme_tidiers), 18
glance.merMod(lme4_tidiers), 15
glance.stanreg(rstanarm_tidiers), 23
glmmADMB_tidiers(glmmadmb_tidiers), 8
glmmadmb_tidiers, 8
glmmTMB_tidiers, 11
insert_NAs, 14
lme4_tidiers, 15
loo.stanfit, 4, 24
loo.stanreg, 24
mad, 24
mcmc_tidiers(tidy.MCMCglmm), 27
na.action, 10, 13, 17, 20
nlme_tidiers, 18
paramNamesMCMCglmm, 23
posterior_interval.stanreg, 23
print.stanreg, 24
ranef.MCMCglmm, 7, 21, 23
raneflevels, 22
rstanarm_tidiers, 23
stan_glmm, 24
stan_glmer, 24
stan_lm, 24
stdranef, 25
summary.lmerModLmerTest, 16
tidy.brmsfit(brms_tidiers), 3
tidy.gamlss(gamlss_tidiers), 7
tidy.glmmadmb(glmmadmb_tidiers), 8
tidy.glmmTMB(glmmTMB_tidiers), 11
tidy.gls(nlme_tidiers), 18
tidy.lme(nlme_tidiers), 18
tidy.mcmc(tidy.MCMCglmm), 27
tidy.MCMCglmm, 27
tidy.merMod(lme4_tidiers), 15
tidy.rjags(tidy.MCMCglmm), 27
tidy.rlmerMod(lme4_tidiers), 15
tidy.stanfit(tidy.MCMCglmm), 27
tidy.stanreg(rstanarm_tidiers), 23
tidy.TMB, 30
tidyMCMC(tidy.MCMCglmm), 27
unrowname, 31