

# Package: bootmlm (via r-universe)

May 13, 2026

**Type** Package

**Title** Bootstrap Resampling for Multilevel Models

**Version** 0.1.1

**Description** Functions for bootstrapping with multilevel data and models (and mixed-effect models). It implements multiple bootstrap methods under the parametric, residual, and case bootstrap categories, as discussed in Van der Leeden, Meijer, and Busing (2008) <[doi:10.1007/978-0-387-73186-5\\_11](https://doi.org/10.1007/978-0-387-73186-5_11)> and Carpenter, Goldstein, and Rasbash (2003) <[doi:10.1111/1467-9876.00415](https://doi.org/10.1111/1467-9876.00415)>. Currently it supports fitted objects from the 'lme4' package.

**URL** <https://github.com/marklhc/bootmlm>,  
<https://marklhc.github.io/bootmlm/>

**BugReports** <https://github.com/marklhc/bootmlm/issues>

**Depends** R (>= 4.1.0)

**Imports** boot (>= 1.3-19), lme4 (>= 1.1-16), Matrix (>= 1.2-11),  
methods, numDeriv, stats, utils

**Suggests** nlme, testthat (>= 3.0.0), MASS, knitr, rmarkdown, haven,  
msm, dplyr, purrr, ggplot2

**LazyData** true

**Config/testthat/edition** 3

**License** MIT + file LICENSE

**Encoding** UTF-8

**VignetteBuilder** knitr

**Config/roxygen2/version** 8.0.0

**NeedsCompilation** no

**Author** Mark Lai [aut, cre, cph]

**Maintainer** Mark Lai <[marklhc@gmail.com](mailto:marklhc@gmail.com)>

**Repository** <https://cran.r-universe.dev>

**Date/Publication** 2026-05-13 13:33:20 UTC

**RemoteUrl** <https://github.com/cran/bootmlm>

**RemoteRef** HEAD

**RemoteSha** 10108546771b7954a8d5b43f5c7c0acaa459e92a

## Contents

bootstrap_mer . . . . .	2
confint.boot . . . . .	4
devfun_mer . . . . .	5
empinf_mer . . . . .	6
pop_syn . . . . .	7
prof_ci_icc . . . . .	8
scores_mer . . . . .	9
solve_eigen_sqrt . . . . .	10
vcov_theta . . . . .	10
vcov_vc . . . . .	11

**Index** **13**

---

bootstrap_mer	<i>Run Various Bootstrap for Mixed Models.</i>
---------------	--

---

## Description

Run multilevel parametric, residual, and case bootstrap with different options

## Usage

```
bootstrap_mer(
  x,
  FUN,
  nsim = 1,
  type = c("parametric", "residual", "residual_cgr", "residual_trans", "reb", "case"),
  corrected_trans = FALSE,
  lv1_resample = FALSE,
  reb_scale = FALSE,
  .progress = FALSE,
  verbose = FALSE,
  ...
)
```

**Arguments**

x	A fitted merMod object from <a href="#">lmer</a> .
FUN	A function taking a fitted merMod object as input and returning the statistic of interest, which must be a (possibly named) numeric vector.
nsim	A positive integer telling the number of simulations, positive integer; the bootstrap $R$ .
type	A character string indicating the type of multilevel bootstrap. Currently, possible values are "parametric", "residual", "residual_cgr", "residual_trans", "reb", or "case".
corrected_trans	Logical indicating whether to use the correct variance-covariance matrix of the residuals. If FALSE, use the variance of $y$ ; if TRUE, use the variance of $y - X\hat{\beta}$ . Only used for type = "residual_trans".
lv1_resample	Logical indicating whether to sample with replacement the level-1 units for each level-2 cluster. Only used for type = "case". Default is FALSE.
reb_scale	Logical indicating whether to scale the residuals for the random effect block bootstrap
.progress	Logical indicating whether to display progress bar (using <a href="#">txtProgressBar</a> ).
verbose	Logical indicating if progress should print output.
...	argument passed to .resid_resample.

**Details**

bootstrap\_mer performs different bootstrapping methods to fitted model objects using the **lme4** package. Currently, only models fitted using [lmer](#) is supported.

**Value**

An object of S3 class "boot", compatible with **boot** package's [boot\(\)](#). It contains the following components:

t0	The original statistic from FUN(x).
t	A matrix with nsim rows containing the bootstrap distribution of the statistic.
R	The value of nsim passed to the function.
data	The data used in the original analysis.
statistic	The function FUN passed to bootstrap_mer.

See the documentation in for `link[boot]{boot}()` for the other components.

**References**

Carpenter, J. R., Goldstein, H., & Rasbash, J. (2003). A novel bootstrap procedure for assessing the relationship between class size and achievement. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 52, 431–443. <https://doi.org/10.1111/1467-9876.00415>

Chambers, R., & Chandra, H. (2013). A random effect block bootstrap for clustered data. *Journal of Computational and Graphical Statistics*, 22(2), 452–470. <https://doi.org/10.1080/10618600.2012.681216>

Davison, A. C. and Hinkley, D. V. (1997). *Bootstrap methods and their application*. Cambridge, UK: Cambridge University Press.

Morris, J. S. (2002). The BLUPs are not "best" when it comes to bootstrapping. *Statistics & Probability Letters*, 56(4), 425–430. [https://doi.org/10.1016/S0167-7152\(02\)00041-X](https://doi.org/10.1016/S0167-7152(02)00041-X)

Van der Leeden, R., Meijer, E., & Busing, F. M. T. A. (2008). Resampling multilevel models. In J. de Leeuw & E. Meijer (Eds.), *Handbook of multilevel Analysis* (pp. 401–433). New York, NY: Springer.

### See Also

- [boot](#) for single-level bootstrapping,
- [bootMer](#) for parametric and semi-parametric bootstrap implemented in lme4, and
- [boot.ci](#) for getting bootstrap confidence intervals and [plot.boot](#) for plotting the bootstrap distribution.

### Examples

```
library(lme4)
fm01ML <- lmer(Yield ~ (1 | Batch), Dyestuff, REML = FALSE)
mySumm <- function(x) {
  c(getME(x, "beta"), sigma(x))
}
# Covariance preserving residual bootstrap
boo01 <- bootstrap_mer(fm01ML, mySumm, type = "residual", nsim = 100)
# Plot bootstrap distribution of fixed effect
library(boot)
plot(boo01, index = 1)
# Get confidence interval
boot.ci(boo01, index = 2, type = c("norm", "basic", "perc"))
# BCa using influence values computed from `empinf_mer`
boot.ci(boo01, index = 2, type = "bca", L = empinf_mer(fm01ML, mySumm, 2))
```

---

confint.boot

*Bootstrap confidence intervals for Two-Level Mixed Models*

---

### Description

This is a wrapper for getting CIs for multiple parameters after running [bootstrap\\_mer](#), to be consistent with a similar method for the [bootMer](#) class.

**Usage**

```
## S3 method for class 'boot'
confint(
  object,
  parm,
  level = 0.95,
  type = c("norm", "basic", "perc", "bca"),
  L = NULL,
  ...
)
```

**Arguments**

object	an object returned by <a href="#">bootstrap_mer</a> .
parm	a specification of which parameters are to be given confidence intervals as a vector of numbers. If missing, all parameters are considered.
level	the confidence level required.
type	character indicating the type of intervals required, as described in <a href="#">boot.ci</a> . Currently "stud" is not supported.
L	empirical influence values required for type = "bca" as described in <a href="#">boot.ci</a> .
...	additional argument(s) passed to <a href="#">boot.ci</a> .

**Value**

A matrix (or vector) with columns giving lower and upper confidence limits for each parameter.

**Examples**

```
library(lme4)
fm01ML <- lmer(Yield ~ (1 | Batch), Dyestuff, REML = FALSE)
mySumm <- function(x) {
  c(getME(x, "beta"), sigma(x))
}

# residual bootstrap
boo_resid <- bootstrap_mer(fm01ML, mySumm, type = "residual", nsim = 100)
confint(boo_resid, type = "bca", L = empinf_merm(fm01ML, mySumm))
```

**Description**

Creates a deviance function for a fitted model object, using  $\theta$  and *sigma* as the parameters.

**Usage**

```
devfun_mer(x)

devfun_mer2(x)
```

**Arguments**

x                    A fitted merMod object from [lmer](#).

**Details**

The built-in function(s) in **lme4** for generating deviance function are not exported and rely on C++ code. This function is mainly used to obtain Hessian and asymptotic covariance matrix of the random effects.

**Value**

A deviance function with one argument `th_sig` that takes input of a numeric vector corresponding to the some estimated values of  $\theta$  and  $\sigma$ . For `devfun_mer2`, a function with one argument `theta` that profiles out  $\sigma$  and only takes input of elements for  $\theta$ .

**References**

Bates, D., Maechler, M., Bolker, B. M., & Walker, S. C. Fitting linear mixed-effects models using lme4. Retrieved from <https://www.jstatsoft.org/article/view/v067i01>  
Implementation in **lme4pureR**: <https://github.com/lme4/lme4pureR/blob/master/R/pls.R>

**Examples**

```
library(lme4)
fm01ML <- lmer(Yield ~ (1 | Batch), Dyestuff, REML = FALSE)
dd <- devfun_mer(fm01ML)
# Asymptotic variance-covariance matrix of (theta, sigma):
2 * solve(numDeriv::hessian(dd, c(fm01ML@theta, sigma(fm01ML))))
```

---

empinf\_mer

*Empirical Influence Values for Two-Level Mixed Models*


---

**Description**

This function calculates the empirical influence values for a statistic in a given fitted model object using the delete- $m_j$  jackknife.

**Usage**

```
empinf_mer(x, FUN, index = 1)

empinf_merm(x, FUN)
```

**Arguments**

x	A fitted merMod object from <a href="#">lmer</a> .
FUN	A function taking a fitted merMod object as input and returning the statistic of interest.
index	An integer stating the position of the statistic in the output of FUN(x).

**Details**

empinf\_mer computes non-parametric influence function of models fitted using [lmer](#) by deleting one cluster at a time. See van der Leeden, Meijer, and Busing (2008, pp. 420–422) for more information. Whereas empinf\_mer computes influence values for a specified position (as specified with the index argument) of the output of FUN, empinf\_merm computes influence values for every element in FUN(x).

**Value**

A numeric vector with length equals to number of clusters of x containing the weighted influence value of each cluster.

**References**

Van der Leeden, R., Meijer, E., & Busing, F. M. T. A. (2008). Resampling multilevel models. In J. de Leeuw & E. Meijer (Eds.), *Handbook of multilevel Analysis* (pp. 401–433). New York, NY: Springer.

**Examples**

```
library(lme4)
fm01ML <- lmer(Yield ~ (1 | Batch), Dyestuff, REML = FALSE)
# Define function for intraclass correlation
icc <- function(x) 1 / (1 + 1 / getME(x, "theta")^2)
empinf_mer(fm01ML, icc)
empinf_mer(fm01ML, fixef)
```

---

pop\_syn

*Synthetic Pupil Popularity Dataset*

---

**Description**

A synthetic two-level dataset with pupils nested within schools, generated to mimic the structure and parameters of the *popular* dataset from Hox (2010). It can be used to demonstrate multilevel bootstrap methods without depending on external data sources.

**Usage**

pop\_syn

**Format**

A data frame with 2000 rows and 5 variables:

**pupil** Pupil identification number within school (integer).

**school** School identification number, 1–100 (integer).

**popular** Pupil popularity score on a 0–10 scale (numeric).

**sex** Pupil sex: 0 = boy, 1 = girl (integer).

**texp** Teacher experience in years (numeric).

**Details**

The dataset was generated by fitting a two-level random intercept model to the original *popular* data and simulating new data from the estimated parameters:

- Fixed effects: intercept = 3.56, sex = 0.84, texp = 0.093
- School-level random intercept SD: 0.69
- Residual SD: 0.68

Continuous predictions were rounded to the nearest integer and clamped to the [0, 10] range to match the original Likert-type scale.

**Source**

Simulated from parameters estimated from the *popular* dataset in: Hox, J. J. (2010). *Multilevel analysis: Techniques and applications* (2nd ed.). Routledge. Data available at [https://stats.oarc.ucla.edu/stat/stata/examples/mlm\\_ma\\_hox/popular.dta](https://stats.oarc.ucla.edu/stat/stata/examples/mlm_ma_hox/popular.dta).

---

prof\_ci\_icc

*Profile Likelihood Confidence Interval for Intraclass Correlation*

---

**Description**

Compute confidence intervals for the intraclass correlation of a model fit of class `merMod-class`.

**Usage**

```
prof_ci_icc(x, level = 0.95, eps_max = 1e+06)
```

**Arguments**

x	A fitted <code>merMod</code> object from <code>lmer</code> .
level	Confidence level between 0 and 1. Default is .95.
eps_max	The maximum value that the upper limit can be. Default is 1e6.

**Details**

This function uses the [uniroot](#) function and determine the lower and upper limit by evaluating the profile deviance (for ML) or the -2 profile REML criterion. It works by obtaining the interval for  $\theta = \tau/\sigma$  and transforming the two limits.

The resulting interval is bounded by zero for its lower limit.

**Value**

A named numeric vector with two values showing the lower and upper limit of the intraclass correlation.

**Examples**

```
library(lme4)
fm01ML <- lmer(Yield ~ (1 | Batch), Dyestuff, REML = FALSE)
# Profile likelihood 95% CI for the ICC
prof_ci_icc(fm01ML)
# 90% CI
prof_ci_icc(fm01ML, level = 0.90)
```

---

 scores\_mer

*Score Functions and Case-wise Derivatives*


---

**Description**

Score Functions and Case-wise Derivatives

**Usage**

```
scores_mer(x, level = 2)
```

**Arguments**

x	A fitted merMod object from <a href="#">lmer</a> .
level	If level = 1, scores at level-1 are returned; if level = 2, which is the default, aggregated scores at the cluster- level are returned.

**Value**

A numeric matrix of score contributions. If level = 2, rows correspond to level-2 clusters (aggregated); if level = 1, rows correspond to level-1 observations. Columns correspond to model parameters in order: fixed effects, variance components, residual variance.

**Examples**

```
library(lme4)
fm01ML <- lmer(Yield ~ (1 | Batch), Dyestuff, REML = FALSE)
# Cluster-level (level-2) scores
scores_mer(fm01ML, level = 2)
# Observation-level (level-1) scores
scores_mer(fm01ML, level = 1)
```

---

solve_eigen_sqrt	<i>Get the square root of a matrix using eigenvalue decomposition, and solve for a linear system</i>
------------------	--

---

**Description**

Solve for  $x$  in the linear system  $Ax = b$ , where  $A$  is a symmetric matrix square root of  $M$  with eigenvalue decomposition such that  $AA = M$ .

**Usage**

```
solve_eigen_sqrt(M, b)
```

**Arguments**

M	a symmetric positive definite/semi-definite matrix
b	a numeric vector or matrix

**Value**

A numeric vector or matrix  $x$  satisfying  $Ax = b$ , where  $A$  is the symmetric square root of  $M$ .

---

vcov_theta	<i>Asymptotic Covariance Matrix for Cholesky Factor of Random Effects</i>
------------	---

---

**Description**

Asymptotic Covariance Matrix for Cholesky Factor of Random Effects

**Usage**

```
vcov_theta(x)
```

**Arguments**

x	A fitted merMod object from <a href="#">lmer</a> .
---	--

**Value**

A symmetric matrix giving the asymptotic covariance matrix of the Cholesky factor  $\theta$  of the random-effects covariance.

**See Also**

[vcov\\_vc](#)

**Examples**

```
library(lme4)
fm01ML <- lmer(Yield ~ (1 | Batch), Dyestuff, REML = FALSE)
# Asymptotic covariance of the Cholesky factor theta
vcov_theta(fm01ML)
```

---

vcov\_vc

*Asymptotic Covariance Matrix for Random Effects*


---

**Description**

Return the asymptotic covariance matrix of random effect standard deviations (or variances) for a fitted model object, using the Hessian evaluated at the (restricted) maximum likelihood estimates.

**Usage**

```
vcov_vc(x, sd_cor = TRUE, print_names = TRUE)
```

**Arguments**

<code>x</code>	A fitted merMod object from <a href="#">lmer</a> .
<code>sd_cor</code>	Logical indicating whether to return asymptotic covariance matrix on SD scale (if TRUE) or on variance scale (if FALSE).
<code>print_names</code>	Logical, whether to print the names for the covariance matrix.

**Details**

Although it's easy to obtain the Hessian for  $\theta$ , the relative Cholesky factor, in **lme4**, there is no easy way to obtain the Hessian for the variance components. This function uses [devfun\\_mer\(\)](#) to obtain the Hessian ( $H$ ) of variance components (or standard deviations, SD), and then obtain the asymptotic covariance matrix as  $-2H^{-1}$ .

**Value**

A  $(q + 1) * (q + 1)$  symmetric matrix of the covariance matrix of  $(\tau, \sigma)$  (if `sd_cor = TRUE`) or  $(\tau^2, \sigma^2)$  (if `sd_cor = FALSE`), where  $q$  is the the number of estimated random-effects components (excluding  $\sigma$ ). For example, for a model with random slope,  $\tau =$  (intercept SD, intercept-slope correlation, slope SD).

**See Also**

[vcov.merMod](#) for covariance matrix of fixed effects, [confint.merMod](#) for confidence intervals of all parameter estimates, and [devfun\\_mer](#) for the underlying function to produce the deviance function.

**Examples**

```
library(lme4)
data(Orthodont, package = "nlme")
fm1 <- lmer(distance ~ age + (age | Subject), data = Orthodont)
vc <- VarCorr(fm1)
# Standard deviation only
print(vc, comp = c("Std.Dev"))
# Asymptotic variance-covariance matrix of (tau, sigma):
vcov_vc(fm1, sd_cor = TRUE)

# Compare with (parametric) bootstrap results :
get_sdcor <- function(x) {
  as.data.frame(lme4::VarCorr(x), order = "lower.tri")[ , "sdcor"]
}
boo <- bootstrap_mer(fm1, get_sdcor, type = "parametric", nsim = 200L)
# There might be failures in some resamples
cov(boo$t, use = "complete.obs")
```

# Index

## \* datasets

pop\_syn, 7

boot, 3, 4

boot.ci, 4, 5

bootMer, 4

bootstrap\_mer, 2, 4, 5

confint.boot, 4

confint.merMod, 12

devfun\_mer, 5, 11, 12

devfun\_mer2 (devfun\_mer), 5

empinf\_mer, 6

empinf\_merm (empinf\_mer), 6

lmer, 3, 6–11

plot.boot, 4

pop\_syn, 7

prof\_ci\_icc, 8

scores\_mer, 9

solve\_eigen\_sqrt, 10

txtProgressBar, 3

uniroot, 9

vcov.merMod, 12

vcov\_theta, 10

vcov\_vc, 11, 11