Package: bootmlm (via r-universe)

May 14, 2026

Type Package

Title Bootstrap Resampling for Multilevel Models

Version 0.1.0.9000

Description An implementation of 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. 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 GPL-3

Encoding UTF-8

VignetteBuilder knitr

Config/roxygen2/version 8.0.0

Config/pak/sysreqs cmake make

Repository https://marklhc.r-universe.dev

Date/Publication 2026-05-05 07:11:40 UTC

RemoteUrl https://github.com/marklhc/bootmlm

RemoteRef HEAD

RemoteSha 100e2390eb0f50cd0eb3f54e67eff60b29149678

Contents

`bootstrap_mer` *Run Various Bootstrap for Mixed Models.*

Description

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

Usage

```
bootstrap_mer(
  x,
  FUN,
  nsim = 1,
  seed = NULL,
 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 `lmer`.
`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.
`seed`	Optional argument to `set.seed`.

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 txtProgressBar).
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.
seed	The value of .Random.seed when bootstrap_mer started to work.
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

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 bootstrap_mer.
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 boot.ci. Currently "stud" is not supported.
L	empirical influence values required for type = "bca" as described in boot.ci.
...	additional argument(s) passed to boot.ci.

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))
```

---

devfun_mer *Deviance Function for Multilevel Models*

Description

Creates a deviance function for a fitted model object, using θ 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 θ and σ. For `devfun_mer2`, a function with one argument `theta` that profiles out σ and only takes input of elements for θ.

References

Bates, D., Maechler, M., Bolker, B. M., & Walker, S. C. Fitting linear mixed-effects models using lme4. Retrieved from https://cran.r-project.org/web/packages/lme4/vignettes/lmer.pdf

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 `lmer`.
`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.

`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 merMod object from `lmer`.
`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.

`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 `lmer`.
`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.

`solve_eigen_sqrt` *Get the square root of a matrix using eigenvalue decomposition, and solve for a linear system*

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

vcov_theta *Asymptotic Covariance Matrix for Cholesky Factor of Random Effects*

Description

Asymptotic Covariance Matrix for Cholesky Factor of Random Effects

Usage

```
vcov_theta(x)
```

Arguments

x	A fitted merMod object from lmer.

See Also

vcov_vc

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

x	A fitted merMod object from lmer.
sd_cor	Logical indicating whether to return asymptotic covariance matrix on SD scale (if TRUE) or on variance scale (if FALSE).
print_names	Logical, whether to print the names for the covariance matrix.

Details

Although it's easy to obtain the Hessian for θ, 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 (τ, σ) (if `sd_cor = TRUE`) or (τ^2, σ^2) (if `sd_cor = FALSE`), where q is the the number of estimated random-effects components (excluding σ). For example, for a model with random slope, τ = (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)

## Not run:
#' # 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")

## End(Not run)
```

Index