Package 'glmmboot'

March 30, 2020

Type Package

Title Bootstrap Resampling for Mixed Effects and Plain Models

Version 0.5.1

Description Performs bootstrap resampling for most models that update() works for. There are two primary functions: bootstrap_model() performs block resampling if random effects are present, and case resampling if not; bootstrap_ci() converts output from bootstrap model runs into confidence intervals and p-values. By default, bootstrap_model() calls bootstrap_ci().
Package motivated by Humphrey and Swingley (2018) <arXiv:1805.08670>.

License AGPL-3 | file LICENSE

URL https://github.com/ColmanHumphrey/glmmboot

Encoding UTF-8

LazyData true

Depends R (>= 3.1)

Imports methods, stats

Suggests glmmTMB (>= 0.2.1), testthat (>= 0.11.0), parallel (>= 3.0.0), future.apply (>= 1.1.0), knitr, rmarkdown, covr

RoxygenNote 7.0.2

VignetteBuilder knitr

NeedsCompilation no

Author Colman Humphrey [aut, cre]

Maintainer Colman Humphrey <humphrc@tcd.ie>

Repository CRAN

Date/Publication 2020-03-30 08:50:02 UTC

R topics documented:

`bootstrap_ci` *Generating bootstrap confidence intervals.*

Description

Enter first level estimates and second level estimates, get bootstrap interval, from the pivotal bootstrap t (Efron and Tibshirani 1994, also endorsed by Hesterberg 2015).

Usage

```
bootstrap_ci(
  base_coef_se = NULL,
  resampled_coef_se = NULL,
  orig_df = NULL,
  alpha_level = 0.05,
  probs = NULL
)

BootCI(
  base_coef_se = NULL,
  resampled_coef_se = NULL,
  orig_df = NULL,
  alp_level = 0.05,
  probs = NULL
)
```

Arguments

`base_coef_se`	Estimates and SEs from full sample. In matrix form, i.e. a $(p+1)x2$ matrix, first column is estimates, second is standard errors. This is the output from using: `coef(summary(model_output))[,1:2,drop = FALSE]` or `coef(summary(model_output))$cond[,1:` `= FALSE]` if `model_output` is the output from a random effects model (some may not have `cond` as the correct pull).
`resampled_coef_se`	List of estimates and SEs from the bootstrapped resamples, each list entry has the same format as the base_coef_se above.
`orig_df`	Degrees of freedom to use to calculate the t-values used for the base interval.
`alpha_level`	level of CI - if you fill in `probs`, will use those instead
`probs`	Default `NULL`, and will use `alpha_level` to set endpoints. Else will calculate these CI endpoints.
`alp_level`	now alpha_level

Value

A matrix containing:

- Estimates
- Bootstrap interval endpoints
- Bootstrap p-value
- Base p-value
- Base interval endpoints
- Relative width of bootstrap interval to base

Examples

```
x <- rnorm(20)
y <- rnorm(20) + x
xy_data = data.frame(x = x, y = y)
first_model <- lm(y ~ x, data = xy_data)
out_list <- bootstrap_model(first_model, base_data = xy_data, 20,
                           return_coefs_instead = TRUE)
bootstrap_ci(out_list$base_coef_se,
             out_list$resampled_coef_se)


  data(test_data)
  library(glmmTMB)
  ## where subj is a random effect
  test_model <- glmmTMB(y ~ x_var1 + (1 | subj),
                        data = test_data, family = binomial)
  output_lists <- bootstrap_model(test_model, base_data = test_data, 199,
                                  return_coefs_instead = TRUE)
  bootstrap_ci(output_lists$base_coef_se,
               output_lists$resampled_coef_se)
```

`bootstrap_model` *Computes bootstrap resamples of your data, stores estimates + SEs.*

Description

By default, this will compute bootstrap resamples and then send them to `bootstrap_ci` for calculation.

Usage

```
bootstrap_model(
  base_model,
  base_data,
  resamples = 9999,
  return_coefs_instead = FALSE,
  parallelism = c("none", "future", "parallel"),
  resample_specific_blocks = NULL,
  unique_resample_lim = NULL,
  narrowness_avoid = TRUE,
  num_cores = NULL,
  future_packages = NULL,
  suppress_sampling_message = !interactive()
)

BootGlmm(
  base_model,
  resamples = 9999,
  base_data = NULL,
  return_coefs_instead = FALSE,
  resample_specific_blocks = NULL,
  unique_resample_lim = NULL,
  narrowness_avoid = TRUE,
  num_cores = NULL,
  suppress_sampling_message = FALSE,
  suppress_loading_bar = FALSE,
  allow_conv_error = FALSE
)
```

Arguments

`base_model` The pre-bootstrap model, i.e. the model output from running a standard model call. Examples: `base_model <-glmmTMB(y ~ age + (1 | subj),data = rel_data,family = binomial)` `base_model <-lm(y ~ x,data = xy_frame)`

`base_data` The data that was used in the call. You can leave this to be automatically read, but I highly recommend supplying it

`resamples` How many resamples of your data do you want to do? 9,999 is a reasonable default (see Hesterberg 2015), but start very small to make sure it works on your data properly, and to get a rough timing estimate etc.

`return_coefs_instead`
Logical, default `FALSE`: do you want the list of lists of results for each bootstrap sample (set to `TRUE`), or the matrix output of all samples? See return for more details.

`parallelism` Type of parallelism (if any) to use to run the resamples. Options are:

- `"none"` The default, sequential
- `"future"` To use `future.apply` (futures)
- `"parallel"` To use `parallel::mclapply`

`resample_specific_blocks`
: Character vector, default `NULL`. If left `NULL`, this algorithm with choose ONE random block to resample over - the one with the largest entropy (often the one with most levels). If you wish to resample over specific random effects as blocks, enter the names here - can be one, or many. Note that resampling multiple blocks is in general quite conservative. If you want to perform case resampling but you DO have random effects, set `resample_specific_blocks` to any non-null value that isn't equal to a random effect variable name.

`unique_resample_lim`
: Should be same length as number of random effects (or left `NULL`). Do you want to force the resampling to produce a minimum number of unique values in sampling? Don't make this too big. Must be named same as rand cols

`narrowness_avoid`
: Boolean, default `TRUE`. If `TRUE`, will resample n-1 instead of n elements in the bootstrap (n being either rows, or random effect levels, depending on existence of random effects). If `FALSE`, will do typical size n resampling.

`num_cores`
: How many cores to use. Defaults to `parallel::detectCores() -1L` if `parallelism = "parallel"`

`future_packages`
: Packages to pass to created futures when using `parallelism = "future"`. This must be supplied if the package used to model the data isn't in base and you're using a plan that doesn't have shared memory, because the model is updated with the S3 generic `update`.

`suppress_sampling_message`
: Logical, the default is to suppress if not in an interactive session. Do you want the function to message the console with the type of bootstrapping? If block resampling over random effects, then it'll say what effect it's sampling over; if case resampling - in which case it'll say as much. Set `TRUE` to hide message.

`suppress_loading_bar`
: defunct now

`allow_conv_error`
: defunct now

Value

By default (with `return_coefs_instead` being `FALSE`), returns the output from `bootstrap_ci`; for each set of covariates (usually just the one set, the conditional model) we get a matrix of output: a row for each variable (including the intercept), estimate, CIs for boot and base, p-values. If `return_coefs_instead` is `TRUE`, then will instead return a list of length two:

- a list containing the output for the base model
- a list of length `resamples` each a list of matrices of estimates and standard errors for each model.

This output is useful for error checking, and if you want to run this function in certain distributed ways.

Examples

```
x <- rnorm(20)
y <- rnorm(20) + x
xy_data = data.frame(x = x, y = y)
first_model <- lm(y ~ x, data = xy_data)

out_matrix <- bootstrap_model(first_model, base_data = xy_data, 20)
out_list <- bootstrap_model(first_model,
                            base_data = xy_data,
                            resamples = 20,
                            return_coefs_instead = TRUE)


  data(test_data)
  library(glmmTMB)
  test_formula <- as.formula('y ~ x_var1 + x_var2 + x_var3 + (1|subj)')
  test_model <- glmmTMB(test_formula, data = test_data, family = binomial)
  output_matrix <- bootstrap_model(test_model, base_data = test_data, 199)

  output_lists <- bootstrap_model(test_model,
                                  base_data = test_data,
                                  resamples = 199,
                                  return_coefs_instead = TRUE)
```

`combine_resampled_lists` *Combines output from multiple bootstrap_model calls*

Description

If you run glmmboot on e.g. a grid of computers, set `return_coefs_instead = TRUE` for each. Then enter them all here. Either just list them out, or put them into one list and enter them.

Usage

```
combine_resampled_lists(..., return_combined_list = FALSE)

CombineResampledLists(..., return_combined_list = FALSE)
```

Arguments

`...`	List of outputs to be combined, or just a bunch of output entries as separate unnamed arguments.
`return_combined_list`	Logical, default `FALSE`. `TRUE` if you want the combined list of lists, `FALSE` for just the output from bootstrap_ci applied to it.

Value

Returns the same output as `bootstrap_ci` by default, or the combined list (as if you had just run bootstrap_model once with all resamples) if `return_combined_list` = `TRUE`

Examples

```
data(test_data)
library(glmmTMB)
## where subj is some RE
test_model <- glmmTMB(y ~ x_var1 + (1 | subj),
                      data = test_data,
                      family = binomial)
output_list1 <- bootstrap_model(
    test_model, base_data = test_data, 99, return_coefs_instead = TRUE)
output_list2 <- bootstrap_model(
    test_model, base_data = test_data, 100, return_coefs_instead = TRUE)
output_list3 <- bootstrap_model(
    test_model, base_data = test_data, 100, return_coefs_instead = TRUE)
combine_resampled_lists(output_list1, output_list2, output_list3)

num_blocks = 10
num_total_resamples = 299
reg_list <- list()
for(i in 1:num_blocks){
    if(i < num_blocks){
        block_resamples = floor((num_total_resamples + 1)/num_blocks)
    } else {
        block_resamples = floor((num_total_resamples + 1)/num_blocks - 1)
    }
    reg_list[[i]] = bootstrap_model(test_model,
                                    base_data = test_data,
                                    resamples = block_resamples,
                                    return_coefs_instead = TRUE,
                                    num_cores = 1) ## increase for parallel
}
boot_ci1 <- combine_resampled_lists(reg_list)
full_list <- combine_resampled_lists(reg_list, return_combined_list = TRUE)
boot_ci2 <- bootstrap_ci(full_list$base_coef_se,
                         full_list$resampled_coef_se)
identical(boot_ci1, boot_ci2)
```

`get_rand` *Takes in a formula with bars and gives back the plain names of the columns*

Description

Takes in a formula with bars and gives back the plain names of the columns

Usage

```
get_rand(form_with_bars)
```

Arguments

`form_with_bars`	A formula used in e.g. `lme4` and similar packages. Typically along the lines: `y ~ age + (1 | school)` etc

Value

A vector of the variables that are treated as random.

Examples

```
get_rand("y ~ age + (1 | school)")
get_rand("y ~ income + (1 | school) + (1 | school:section)")
get_rand("y ~ income + (1 | school) + (1 | school/section)")
get_rand(as.formula("y ~ x + (1 | z)"))
get_rand("y ~ x")
```

`test_data` *Simulated data containing three fixed effects and one random effect*

Description

A small normal dataset with a proportional outcome to illustrate the use of this package. The outcome has mean `expit(0.5 + 0.1 * x_var1 + 0.2 * x_var2 + 0.3 * x_var3 + SUBJ_VAL)` where SUBJ_VAL are the values of the random effect The SD of y is then shrunk by 0.9 relative to a binomial distribution, and then beta values are generated. Arbitrarily close to the endpoints gives zeros and ones.

Usage

```
test_data
```

Format

A data frame with 300 rows and 4 variables:

x_var1 independent normally distributed variable

x_var2 independent normally distributed variable

x_var3 independent normally distributed variable

subj levels of random effect

y outcome: lives in interval [0,1]

Index