Package ‘wildmeta’

March 8, 2023

Title  Cluster Wild Bootstrapping for Meta-Analysis
Version  0.3.2
Description  Conducts single coefficient tests and multiple-contrast hypothesis tests of meta-
regression models using cluster wild bootstrapping, based on methods examined in Joshi, Pustejovsky, and Beretvas (2022) <DOI:10.1002/jrsm.1554>.
BugReports  https://github.com/meghapsimatrix/wildmeta/issues
License  GPL-3
Encoding  UTF-8
RoxygenNote  7.2.3
Imports  clubSandwich (>= 0.5.4), sandwich, robumeta, metafor, stats
Suggests  testthat (>= 3.0.0), knitr, rmarkdown, ggplot2, covr,
mockery, future, future.apply, parallelly
VignetteBuilder  knitr
Config/testthat/edition  3
NeedsCompilation  no
Author  Megha Joshi [aut, cre] (<https://orcid.org/0000-0001-7936-076X>),
James Pustejovsky [aut] (<https://orcid.org/0000-0003-0591-9465>),
Pierce Cappelli [aut] (<https://orcid.org/0000-0001-8426-8132>)
Maintainer  Megha Joshi <megha.j456@gmail.com>
Depends  R (>= 3.5.0)
Repository  CRAN
Date/Publication  2023-03-08 17:00:02 UTC

R topics documented:

plot.Wald_test_wildmeta ...................................................... 2
run_cwb ........................................................................ 3
Wald_test_cwb ................................................................. 4

Index  7
plot.Wald_test_wildmeta

Plot distribution of bootstrap test statistics

Description

Creates a density plot showing the distribution of bootstrap test statistics.

Usage

## S3 method for class 'Wald_test_wildmeta'
plot(x, ...)

Arguments

x                   Results from Wald_test_cwb function
...                 Any other arguments to be passed to ggplot2::geom_density()

Value

A ggplot2 density plot.

Examples

data("SATcoaching", package = "clubSandwich")
library(clubSandwich)
library(robumeta)

full_model <- robu(d ~ 0 + study_type + hrs + test,
                  studynum = study,
                  var.eff.size = V,
                  small = FALSE,
                  data = SATcoaching)

res <- Wald_test_cwb(full_model = full_model,
                      constraints = constrain_equal(1:3),
                      R = 99)

if (requireNamespace("ggplot2", quietly = TRUE)) {
  plot(res, fill = "darkred", alpha = 0.5)
}
**run_cwb**

**Calculate bootstrap outcomes or test statistics using cluster wild bootstrapping**

**Description**

Calculate bootstrap outcomes or test statistics using cluster wild bootstrapping for meta-analytic models fit using `robmeta::robu()`, `metafor::rma.mv()`, and `metafor::rma.uni()`.

**Usage**

```r
textwrap::wrap(run_cwb(model, cluster, R, f = NULL, ...,
auxiliary_dist = "Rademacher",
adjust = "CR0",
simplify = FALSE,
seed = NULL,
future_args = NULL,
future_f_args = NULL)
```

**Arguments**

- `model`: Fitted `robmeta::robu()`, `metafor::rma.mv()`, or `metafor::rma.uni()` model. For cluster wild bootstrapping, a null model is recommended, with null model indicating a model containing all variables except the ones being tested.

- `cluster`: Vector indicating which observations belong to the same cluster.

- `R`: Number of bootstrap replications.

- `f`: Optional function to be used to calculate bootstrap test statistics based on the bootstrapped outcomes. If `f` is `NULL` (the default), this function returns a list containing bootstrapped outcomes.

- `...`: Optional arguments to be passed to the function specified in `f`.

- `auxiliary_dist`: Character string indicating the auxiliary distribution to be used for cluster wild bootstrapping, with available options: "Rademacher", "Mammen", "Webb six", "uniform", "standard normal". The default is set to "Rademacher." We recommend the Rademacher distribution for models that have at least 10 clusters. For models with less than 10 clusters, we recommend the use of "Webb six" distribution.

- `adjust`: Character string specifying which small-sample adjustment should be used to multiply the residuals by. The available options are "CR0", "CR1", "CR2", "CR3", or "CR4", with a default of "CR0".
simplify Logical, with TRUE indicating the bootstrapped outcomes or F statistics will be simplified to a vector or matrix and FALSE (the default) indicating the results will be returned as a list.

seed Optional seed value to ensure reproducibility.

future_args Optional list of additional arguments passed to the future_*() functions used in calculating results across bootstrap replications. Ignored if the future.apply package is not available.

future_f_args Optional list of additional arguments passed to the future_*() function used in calculating f results (but not bootstrap outcome vectors) across bootstrap replications. Ignored if the future.apply package is not available.

Value

A list or matrix containing either the bootstrapped outcomes or bootstrapped test statistics.

Examples

library(clubSandwich)
library(robumeta)

model <- robu(d ~ 0 + study_type + hrs + test,
              studynum = study,
              var.eff.size = V,
              small = FALSE,
              data = SATcoaching)

bootstraps <- run_cwb(
  model = model,
  cluster = model$data.full$study,
  R = 12,
  adjust = "CR2",
  simplify = FALSE
)

bootstraps

Wald_test_cwb Calculate p-values with cluster wild bootstrapping for meta-regression models.

Description

Calculate p-values for single coefficient and multiple contrast hypothesis tests using cluster wild bootstrapping.
Wald_test_cwb

Usage

```
Wald_test_cwb(
  full_model,
  constraints,
  R,
  cluster = NULL,
  auxiliary_dist = "Rademacher",
  adjust = "CR0",
  type = "CR0",
  test = "Naive-F",
  seed = NULL,
  future_args = NULL
)
```

Arguments

- **full_model**: Model fit using `robumeta::robu()`, `metafor::rma.mv()`, or `metafor::rma.uni()` that includes the full set of moderators in the meta-regression model.

- **constraints**: A q X p constraint matrix be tested. Alternately, a function to create such a matrix, specified using `clubSandwich::constrain_equal()` or `clubSandwich::constrain_zero()`.

- **R**: Number of bootstrap replications.

- **cluster**: Vector of identifiers indicating which observations belong to the same cluster. If NULL (the default), then the clustering variable will be inferred based on the structure of `full_model`.

- **auxiliary_dist**: Character string indicating the auxiliary distribution to be used for cluster wild bootstrapping, with available options: "Rademacher", "Mammen", "Webb six", "uniform", "standard normal". The default is set to "Rademacher." We recommend the Rademacher distribution for models that have at least 10 clusters. For models with less than 10 clusters, we recommend the use of "Webb six" distribution.

- **adjust**: Character string specifying which small-sample adjustment should be used to multiply the residuals by. The available options are "CR0", "CR1", "CR2", "CR3", or "CR4", with a default of "CR0".

- **type**: Character string specifying which small-sample adjustment is used to calculate the Wald test statistic. The available options are "CR0", "CR1", "CR2", "CR3", or "CR4", with a default of "CR0".

- **test**: Character string specifying which (if any) small-sample adjustment is used in calculating the test statistic. Default is "Naive-F", which does not make any small-sample adjustment.

- **seed**: Optional seed value to ensure reproducibility.

- **future_args**: Optional list of additional arguments passed to the `future_*()` functions used in calculating results across bootstrap replications. Ignored if the future.apply package is not available.
Wald_test_cwb

Value

A data.frame containing the name of the test, the adjustment used for the bootstrap process, the type of variance-covariance matrix used, the type of test statistic, the number of bootstrap replicates, and the bootstrapped p-value.

Examples

```r
library(clubSandwich)
library(robumeta)

model <- robu(d ~ 0 + study_type + hrs + test,
              studynum = study,
              var.eff.size = V,
              small = FALSE,
              data = SATcoaching)

C_mat <- constrain_equal(1:3, coefs = coef(model))

Wald_test_cwb(full_model = model,
               constraints = C_mat,
               R = 12)

# Equivalent, using constrain_equal()
Wald_test_cwb(full_model = model,
              constraints = constrain_equal(1:3),
              R = 12)
```
Index

plot.Wald_test_wildmeta, 2
run_cwb, 3
Wald_test_cwb, 4