Package 'resilience'

May 9, 2026

Type Package

Title Predictors of Resilience to a Stressor in a Single-Arm Study

Version 2025.1.1

Date 2025-12-13

Maintainer Ravi Varadhan <ravi.varadhan@jhu.edu>

Description Studies of resilience in older adults employ a single-arm design where everyone experiences the stressor. The simplistic approach of regressing change versus baseline yields biased estimates due to regression-to-the-mean. This package provides a method to correct the bias. It also allows covariates to be included. The method implemented in the package is described in Varadhan, R., Zhu, J., and Bandeen-Roche, K (2024), Biostatistics 25(4): 1094-1111.

Depends R(>= 4.0.0), parallel, doParallel, foreach

Imports stats, mice, nptest

License GPL (>= 2)

LazyLoad yes

NeedsCompilation no

Repository CRAN

Author Ravi Varadhan [aut, cre],
Jiafeng Zhu [ctb]

RoxygenNote 7.3.3

Encoding UTF-8

LazyData true

Date/Publication 2025-12-15 08:40:02 UTC

Contents

`plot.prepost_mi` *Plot Results from Multiply Imputed Resilience Analysis*

Description

Plot method for visualizing imputation diagnostics

Usage

```
## S3 method for class 'prepost_mi'
plot(x, type = "coefficients", ...)
```

Arguments

`x`	An object of class `prepost_resilience_mi_list`.
`type`	Type of plot: `"estimates"` (default) or `"fmi"`.
`...`	Additional arguments passed to plotting functions.

`prepost` *Identifying Predictors of Resilience to Stressors in Single-Arm Studies of Pre-Post Change*

Description

Studies of resilience in older adults are typically conducted with a single-arm where everyone experiences the stressor. The simplistic approach of regressing change versus baseline yields biased estimates due to mathematical coupling and regression-to-the-mean. This function provides a method to correct the bias.

Usage

```
prepost(
  formula,
  data,
  change = TRUE,
  k = c(1, 1.5, 2),
  m = 1,
  nboot = 1000,
  ci.level = 0.95,
  boot.method = c("perc", "norm", "basic", "bca"),
  ncores = 2
)
```

Arguments

formula	Formula object where LHS is Post response variable and RHS is pre-response variable plus the covariates. Note: the first variable of the RHS of the formula must be the pre-response variable. For example, 'y2 ~ y1 + x1 + x2'.
data	Data frame containing all the variables. Only complete cases are used in the analysis, i.e. rows of dataframe with missing values in any of the predictors are automatically deleted.
change	A logical variable. If 'TRUE' (default) the dependent variable of regression is the pre-post change. If 'FALSE', the post response is used as the dependent variable.
k	A sensitivity analysis parameter. Typically, it is greater than or equal to 1.0. It is recommended that the user provide at least two values to examine how the effects vary with 'k'. Default setting allows three values: k = 1.0, 1.5, and 2.0. For more details about this parameter refer to the manuscript.
m	Another sensitivity analysis parameter. It is set equal to 1.0. It is recommended that the user not change this unless there is information from an external study to justify a different value.
nboot	Number of bootstrap samples for calculating the confidence intervals of corrected regression coefficients. Default is 1000.
ci.level	Confidence coefficient for confidence interval. Default is 0.95 (95% confidence intervals).
boot.method	The bootstrap method for confidence interval. Four options are provided: "perc" (percentile), "norm" (normal approximation), "basic", and "bca" (bias-corrected accelerated bootstrap). Default is "perc".
ncores	Number of cores available for parallel computing. Default is set to 2 due to CRAN requirements. If more cores are available, the user can utilize all available cores with the command: 'ncores = parallel::detectCores()'

Value

A list with the following components:

`naive.beta` Unadjusted, naive estimates of regression coefficients

`corrected.beta` The corrected coefficients of the variables. A matrix with one column of parameter values for each value of sensitivity parameter `k`

`CI` A list of length equal to the number of sensitivity values. Each component of the list is a matrix with two columns of lower and upper confidence interval for each parameter

Missing Data

This function uses complete-case analysis only. If your data contains missing values, the function will issue a warning and exclude cases with any missing values. For more robust inference with missing data, consider using multiple imputation followed by `prepost_resilience_mi_list()`, which pools results across multiply imputed datasets using Rubin's rules.

Parallel Processing

The function uses the `parallel` and `foreach` packages to perform parallel computations of bootstrap confidence intervals for different values of the sensitivity parameter, `k`.

Author(s)

Ravi Varadhan

References

Varadhan, R., Zhu, J., and Bandeen-Roche, K (2024). Identifying Predictors of Resilience to Stressors in Single-Arm Studies of Pre-Post Change. *Biostatistics*. 25(4): 1094-1111.

See Also

`prepost_mi` for analysis with multiply imputed data.

Examples

```
## Not run:
data(tkr)

names(tkr.dat)
dim(tkr.dat)

# pre-post change regression
ans1 <- prepost(post.Y ~ pre.Y + I(age-mean(age)) + I((age - mean(age))^2) +
  bmi + gender + as.factor(smoker), data=tkr.dat, k=c(1.2, 1.5), nboot=200)
print(ans1)

# Post regression
ans2 <- prepost(post.Y ~ pre.Y + I(age-mean(age)) + I((age - mean(age))^2) +
  bmi + gender + as.factor(smoker), data=tkr.dat,
  k=c(1.2, 1.5), change=FALSE, nboot=200, boot.method="norm")
print(ans2)

# without any covariates
ans3 <- prepost(post.Y ~ pre.Y, data=tkr.dat, k=c(1.2, 2.0), nboot=200)
print(ans3)

# Bootstrapping using "bca" - relatively slow
# Not run
# ans4 <- prepost(post.Y ~ pre.Y, data=tkr.dat, k=c(1.2, 2.0), change=FALSE,
#   boot.method = "bca")

## End(Not run)
```

prepost_mi *Prepost Analysis with Pre-Imputed Datasets*

Description

This function takes a list of already imputed datasets and runs prepost (resilience) regression analysis on each, then pools the results using Rubin's rules. Users are responsible for the imputation process, allowing maximum flexibility in imputation methods.

Usage

```
prepost_mi(
  data_list,
  formula,
  k = 1,
  nboot = 500,
  pool_method = "detailed",
  coef_labels = NULL,
  verbose = TRUE,
  ...
)
```

Arguments

data_list	List of imputed datasets (data frames) or a 'mids' object from 'mice'
formula	Formula for the prepost model (e.g., post_score ~ pre_score + age + bmi)
k	Resilience parameter (default = 1.2)
nboot	Number of bootstrap samples (default = 200)
pool_method	Pooling method: "detailed" (Barnard & Rubin) or "simple" (default = "detailed")
coef_labels	Optional named vector for renaming coefficients (e.g., c("age" = "Age (years)"))
verbose	Whether to print progress messages (default = TRUE)
...	Additional arguments passed to prepost()

Value

A list containing:

pooled_results The pooled estimates

results_table Formatted results table

individual_results List of results from each imputation

model_info Information about the model specification

imputation_info Summary of imputation characteristics

See Also

prepost for complete-case analysis without missing data mice for multiple imputation

Examples

```
## Not run:
# Example 1: List of data frames from any imputation method
library(mice)
data <- data.frame(
  post_score = rnorm(100),
  pre_score = rnorm(100),
  age = rnorm(100, 50, 10),
  bmi = rnorm(100, 25, 5)
)

# Create some missing data
data$age[1:10] <- NA
data$bmi[5:15] <- NA

# User does their own imputation (could be from any package/method)
# Method 1: Using mice
imp_mice <- mice(data, m = 5, printFlag = FALSE)
imp_list <- list()
for (i in 1:5) {
  imp_list[[i]] <- complete(imp_mice, i)
}

# Method 2: Using Amelia
if (requireNamespace("Amelia", quietly = TRUE)) {
  imp_amelia <- Amelia::amelia(data, m = 5)
  imp_list <- imp_amelia$imputations
}

# Method 3: Using aregImpute from Hmisc
if (requireNamespace("Hmisc", quietly = TRUE)) {
  set.seed(123)
 imp_areg <- Hmisc::aregImpute(~ post_score + pre_score + age + bmi, data = data, n.impute = 5)
  imp_list <- list()
  for (i in 1:5) {
    imp_data <- data
    imp_data$age[is.na(imp_data$age)] <- imp_areg$imputed$age[, i]
    imp_data$bmi[is.na(imp_data$bmi)] <- imp_areg$imputed$bmi[, i]
    imp_list[[i]] <- imp_data
  }
}

# Run resilience analysis on the pre-imputed list
result <- prepost_mi(
  data_list = imp_list,
  formula = post_score ~ pre_score + age + bmi,
  k = 1.2,
  nboot = 200
```

```
)

# Example 2: Directly using a mice mids object
result2 <- prepost_mi(
  data_list = imp_mice,  # mids object
  formula = post_score ~ pre_score + age + bmi
)

## End(Not run)
```

`print.prepost_mi` *Print Results from Multiply Imputed Resilience Analysis*

Description

S3 print method for prepost_resilience_mi_list

Usage

```
## S3 method for class 'prepost_mi'
print(x, ...)
```

Arguments

`x`	An object of class `prepost_resilience_mi_list`.
`...`	Additional arguments passed to plotting functions.

`summary.prepost_mi` *Summary Results from Multiply Imputed Resilience Analysis*

Description

S3 summary method for prepost_resilience_mi_list

Usage

```
## S3 method for class 'prepost_mi'
summary(object, ...)
```

Arguments

`object`	An object of class `prepost_resilience_mi_list`.
`...`	Additional arguments passed to plotting functions.

tkr.dat *Example dataset for pre-post resilience analysis*

Description

A simulated dataset from a single-arm study with pre/post measurements.

Usage

```
tkr.dat
```

Format

A data frame with columns:

post.Y Post-stressor outcome

pre.Y Baseline outcome

age Age in years

bmi Body mass index

gender Gender (M/F)

smoker Smoking status

Source

Simulated for demonstration.

Index