Package 'IJSE'

September 26, 2024

Type Package

Title Infinite-Jackknife-Based Standard Errors for 'brms' Models

Version 0.1.1

Description Provides a function to calculate infinite-jackknife-based standard errors for fixed effects parameters in 'brms' models, handling both clustered and independent data.
References: Ji et al. (2024) <doi:10.48550/arXiv.2407.09772>; Giordano et al. (2024) <doi:10.48550/arXiv.2305.06466>.

License MIT + file LICENSE

Depends R (>= 3.5.0)

Imports brms, posterior

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.2.3

NeedsCompilation no

Author Feng Ji [aut, cre] (<https://orcid.org/0000-0002-2051-5453>),
JoonHo Lee [aut],
Sophia Rabe-Hesketh [aut]

Maintainer Feng Ji <f.ji@utoronto.ca>

Repository CRAN

Date/Publication 2024-09-26 21:00:53 UTC

Contents

IJ_se *Calculate Infinite-Jackknife-Based Standard Errors for brms Models*

Description

Computes infinite-jackknife-based standard errors for fixed effects parameters from a ‘brmsfit‘ model object. The function handles both clustered and independent data.

Usage

```
IJ_se(fit, cluster_var = NULL)
```

Arguments

`fit`	A ‘brmsfit‘ object resulting from fitting a model using the ‘brms‘ package.
`cluster_var`	An optional vector indicating the cluster membership for each observation. If ‘NULL‘, the function treats the data as independent.

Value

A named vector of standard errors for the fixed effects parameters.

Examples

```
# Load libraries

library(brms)

# Set a seed for reproducibility

set.seed(42)

### Model 1: Linear Regression using brms

# Simulate data

n <- 300
age <- rnorm(n, mean = 40, sd = 10)
income <- rnorm(n, mean = 50000, sd = 10000)
education_years <- rnorm(n, mean = 12, sd = 2)

# True coefficients

beta_0 <- 50000    # Intercept
beta_age <- -1000  # Age effect
beta_income <- 0.5 # Income effect
beta_edu <- 2000   # Education effect
sigma <- 10000     # Residual standard deviation
```

```
# Simulate house prices

house_price <- beta_0 + beta_age * age + beta_income * income +
  beta_edu * education_years + rnorm(n, mean = 0, sd = sigma)

# Create data frame

data_linear <- data.frame(house_price, age, income, education_years)

# Fit the model

fit_linear <- brm(
  formula = house_price ~ age + income + education_years,
  data = data_linear,
  family = gaussian(),
  seed = 42
)

# Summary

summary(fit_linear)

# Obtain IJ-based SE

IJ_se(fit_linear)

### Model 2: Linear Regression for Clustered Data using brms

# Simulate data

n_schools <- 30
students_per_school <- 100
n <- n_schools * students_per_school

# School IDs and types

school_id <- rep(1:n_schools, each = students_per_school)
school_type <- rep(sample(c("Public", "Private"), n_schools, replace = TRUE),
                   each = students_per_school)
school_type_num <- ifelse(school_type == "Public", 0, 1)

# Random intercepts for schools

sigma_school <- 6
u_school <- rnorm(n_schools, mean = 0, sd = sigma_school)
u_school_long <- rep(u_school, each = students_per_school)

# Student-level predictors

student_age <- rnorm(n, mean = 15, sd = 1)
math_score <- rnorm(n, mean = 50, sd = 10)

# True coefficients
```

```
beta_0 <- 50             # Fixed intercept
beta_age <- 1.5          # Age effect
beta_math <- 1           # Math score effect
beta_school_type <- 5    # School type effect
sigma_student <- 3       # Residual standard deviation

# Simulate reading scores

reading_score <- beta_0 + beta_age * student_age + beta_math * math_score +
  beta_school_type * school_type_num + u_school_long +
  rnorm(n, mean = 0, sd = sigma_student)

# Create data frame

data_clustered <- data.frame(
  reading_score,
  student_age,
  math_score,
  school_id = factor(school_id),
  school_type,
  student_id = 1:n
)

# Fit the model

fit_clustered <- brm(
  formula = reading_score ~ student_age + math_score + school_type,
  data = data_clustered,
  family = gaussian(),
  seed = 42
)

# Summary

summary(fit_clustered)

# Obtain IJ-based SE, taking the clustering into account
IJ_se(fit_clustered, cluster_var = data_clustered$school_id)

### Example 3: Quantile Regression using brms

# Independent data for quantile regression
N <- 100
x <- runif(N)
eps <- 1 * x^2 + sin(rchisq(N, 8)) + sin(rnorm(x, 3))  # some random DGP
y <- 2 * x + runif(N) + eps^2

# Create data frame
data_quantile <- data.frame(y, x)

# Fit quantile regression model
fit_quantile <- brm(
```

```
  formula = bf(y ~ x, quantile = .3),  # Quantile regression with 30th percentile
  data = data_quantile,
  family = asym_laplace(link_quantile = "identity"),
  seed = 42
)

# Summary of quantile regression model
summary(fit_quantile)

# Obtain IJ-based SE
IJ_se(fit_quantile)

### Example 4: Quantile Regression for Clustered Data using brms

# Clustered data for quantile regression
J <- 30  # Number of clusters
I <- 50  # Cluster size
subj <- rep(1:J, each = I)
rho <- 0.8

# Random effect and error terms
U <- rnorm(J * I, sd = sqrt(1 / 3))
Z <- rep(rnorm(J, sd = 5), each = I)
E <- rnorm(J * I)

# Covariates and response variable
X <- sqrt(rho) * Z + sqrt(1 - rho) * E
X2 <- X^2
Y <- 0.1 * U + X + X2 * U

# Create data frame
data_cluster_quantile <- data.frame(Y, X, X2, subj = factor(subj))

# Fit quantile regression model
fit_quantile_cluster <- brm(
  formula = bf(Y ~ X + X2, quantile = .33),  # Quantile regression with 33rd percentile
  data = data_cluster_quantile,
  family = asym_laplace(link_quantile = "identity"),
  seed = 42
)

# Summary of quantile regression model
summary(fit_quantile_cluster)

# Obtain IJ-based SE, taking clustering into account
IJ_se(fit_quantile_cluster, cluster_var = data_cluster_quantile$subj)
```

Index