Package: pprof (via r-universe)

October 29, 2026

Type Package

Title Modeling, Standardization and Testing for Provider Profiling

Version 1.0.3

Date 2026-02-08

Description Implements linear and generalized linear models for provider profiling, incorporating both fixed and random effects. For large-scale providers, the linear profiled-based method and the SerBIN method for binary data reduce the computational burden. Provides post-modeling features, such as indirect and direct standardization measures, hypothesis testing, confidence intervals, and post-estimation visualization. For more information, see Wu et al. (2022) <doi:10.1002/sim.9387>.

License MIT + file LICENSE

LazyData true

Imports Rcpp, RcppParallel, stats, caret, olsrr, pROC, poibin, dplyr, ggplot2, Matrix, lme4, magrittr, scales, tibble, rlang, tidyselect, globals

LinkingTo Rcpp, RcppArmadillo, RcppParallel

Depends R (>= 4.1.0)

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

RoxygenNote 7.3.3

URL https://github.com/UM-KevinHe/pprof

BugReports https://github.com/UM-KevinHe/pprof/issues

Encoding UTF-8

SystemRequirements GNU make

Config/testthat/edition 3

NeedsCompilation yes

Author Xiaohan Liu [aut, cre], Lingfeng Luo [aut], Yubo Shao [aut], Xiangeng Fang [aut], Wenbo Wu [aut], Kevin He [aut]

Maintainer Xiaohan Liu <xhliuu@umich.edu>

Config/pak/sysreqs make

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

Date/Publication 2026-02-10 03:32:20 UTC

RemoteUrl https://github.com/cran/pprof

RemoteRef HEAD

RemoteSha 1efc4040de386900a7eeed222e478b18b77a6ff2

Contents

bar_plot . . . 3
caterpillar_plot . . . 4
confint.linear_cre . . . 6
confint.linear_fe . . . 8
confint.linear_re . . . 9
confint.logis_cre . . . 10
confint.logis_fe . . . 12
confint.logis_re . . . 14
data_check . . . 15
ecls_data . . . 16
ExampleDataBinary . . . 17
ExampleDataLinear . . . 18
linear_cre . . . 19
linear_fe . . . 21
linear_re . . . 23
logis_cre . . . 26
logis_fe . . . 28
logis_firth . . . 31
logis_re . . . 34
plot.linear_fe . . . 36
plot.logis_fe . . . 38
SM_output . . . 40
SM_output.linear_cre . . . 40
SM_output.linear_fe . . . 41
SM_output.linear_re . . . 43
SM_output.logis_cre . . . 44
SM_output.logis_fe . . . 46
SM_output.logis_re . . . 47
summary.linear_cre . . . 49
summary.logis_cre . . . 50
summary.logis_fe . . . 52
test . . . 53
test.linear_cre . . . 53
test.linear_fe . . . 55
test.linear_re . . . 56
test.logis_cre . . . 57

test.logis_fe 59
test.logis_re 61

Index **63**

`bar_plot` *Get a bar plot for flagging percentage overall and stratified by provider sizes*

Description

Generate a bar plot for flagging percentage.

Usage

```
bar_plot(
  flag_df,
  group_num = 4,
  bar_colors = c("#66c2a5", "#fc8d62", "#8da0cb"),
  bar_width = 0.7,
  label_color = "black",
  label_size = 4
)
```

Arguments

`flag_df`	a data frame from `test` function containing the flag of each provider.
`group_num`	number of groups into which providers are divided based on their sample sizes. The default is 4.
`bar_colors`	a vector of colors used to fill the bars representing the categories. The default is c("#66c2a5", "#fc8d62", "#8da0cb").
`bar_width`	width of the bars in the bar chart. The default is 0.7.
`label_color`	color of the text labels inside the bars. The default is "black".
`label_size`	size of the text labels inside the bars. The default is 4.

Details

This function generates a bar chart to visualize the percentage of flagging results based on provider sizes. The input data frame `test_df` must be the output from package `pprof`'s `test` function. Providers are grouped into a specified number of groups (`group_num`) based on their sample sizes, where the number of providers are approximately equal across groups. An additional "overall" group is included to show the flagging results across all providers.

Value

A ggplot object representing the bar chart of flagging results.

See Also

test.linear_fe, test.linear_re, test.logis_fe

Examples

```
data(ExampleDataLinear)
outcome <- ExampleDataLinear$Y
covar <- ExampleDataLinear$Z
ProvID <- ExampleDataLinear$ProvID
fit_linear <- linear_fe(Y = outcome, Z = covar, ProvID = ProvID)
test_linear <- test(fit_linear)
bar_plot(test_linear)

data(ExampleDataBinary)
fit_logis <- logis_fe(Y = ExampleDataBinary$Y,
                      Z = ExampleDataBinary$Z,
                      ProvID = ExampleDataBinary$ProvID, message = FALSE)
test_logis <- test(fit_logis)
bar_plot(test_logis)
```

---

caterpillar_plot *Get a caterpillar plot to display confidence intervals for standardized measures*

Description

Generate a caterpillar plot for standardized measures from different models using a provided CI dataframe.

Usage

```
caterpillar_plot(
  CI,
  point_size = 2,
  point_color = "#475569",
  refline_value = NULL,
  refline_color = "#64748b",
  refline_size = 1,
  refline_type = "dashed",
  errorbar_width = 0,
  errorbar_size = 0.5,
  errorbar_alpha = 0.5,
  errorbar_color = "#94a3b8",
  use_flag = FALSE,
  orientation = "vertical",
  flag_color = c("#E69F00", "#56B4E9", "#009E73")
)
```

Arguments

CI	a dataframe from confint function containing the standardized measure values, along with their confidence intervals lower and upper bounds.
point_size	size of the points in the caterpillar plot. The default value is 2.
point_color	color of the points in the plot. The default value is "#475569".
refline_value	value of the horizontal reference line, for which the standardized measures are compared. The default value is NULL.
refline_color	color of the reference line. The default value is "#64748b".
refline_size	size of the reference line. The default value is 1.
refline_type	line type for the reference line. The default value is "dashed".
errorbar_width	the width of the error bars (horizontal ends of the CI bars). The default value is 0.
errorbar_size	the thickness of the error bars. The default value is 0.5.
errorbar_alpha	transparency level for the error bars. A value between 0 and 1, where 0 is completely transparent and 1 is fully opaque. The default value is 0.5.
errorbar_color	color of the error bars. The default value is "#94a3b8".
use_flag	logical; if TRUE, the error bars are colored to show providers' flags based on their performance. The default is FALSE.
orientation	a string specifies the orientation of the caterpillar plot: **"vertical"** (default) providers on the x-axis and values on the y-axis. **"horizontal"** providers on the y-axis and values on the x-axis.
flag_color	vector of colors used for flagging providers when use_flag = TRUE. The default value is c("#E69F00", "#56B4E9", "#009E73").

Details

This function creates a caterpillar plot to visualize the standardized measures (indirect or direct). The input CI must be a dataframe output from package pprof's confint function. Each provider's standardized measure value is represented as a point, and a reference line is shown at the value specified by refline_value (default is NULL). If refline_value is not specified, for linear FE or RE models with indirect or direct standardized differences, it will be set to 0; for logistic FE models with indirect or direct ratios, it will be set to 1; and for logistic FE with indirect or direct rates, it will be set to the population rate, which represents the average rate across all observations.

Confidence intervals (CI) are displayed as error bars: for alternative = "two.sided", two-sided confidence intervals are shown; for alternative = "greater", error bars extend from the lower bound to the standardized measure values; and for alternative = "less", they extend from the standardized measure values to the upper bound. For cases where one side of the confidence interval is infinite, that side only extends to the standardized measure. For example, in a logistic fixed effect model, if a provider has all 0s or all 1s, one side of the confidence interval will be infinite.

When use_flag = TRUE, the plot will use colors specified by flag_color to show the flags of providers. Each error bar will be colored to reflect the flag, making it easy to identify providers with different performance levels. When use_flag = FALSE, all error bars will have the same color, specified by errorbar_color. This provides a simpler visualization without flagging individual providers.

Value

A ggplot object which is a caterpillar plot for the standardized measures.

See Also

confint.linear_fe, confint.linear_re, confint.logis_fe

Examples

```
data(ExampleDataLinear)
outcome <- ExampleDataLinear$Y
covar <- ExampleDataLinear$Z
ProvID <- ExampleDataLinear$ProvID
fit_linear <- linear_fe(Y = outcome, Z = covar, ProvID = ProvID)
CI_linear <- confint(fit_linear)
caterpillar_plot(CI_linear$CI.indirect, use_flag = TRUE,
                 errorbar_width = 0.5, errorbar_size = 1)

data(ExampleDataBinary)
fit_logis <- logis_fe(Y = ExampleDataBinary$Y,
                      Z = ExampleDataBinary$Z,
                      ProvID = ExampleDataBinary$ProvID, message = FALSE)
CI_logis <- confint(fit_logis)
caterpillar_plot(CI_logis$CI.indirect_ratio, use_flag = TRUE,
                 errorbar_width = 0.5, errorbar_size = 1,
                 orientation = "horizontal")
```

---

confint.linear_cre *Get confidence intervals for provider effects or standardized measures from a fitted* linear_cre *object*

Description

Provide confidence intervals for provider effects or standardized measures from a correlated random effect linear model.

Usage

```
## S3 method for class 'linear_cre'
confint(
  object,
  parm,
  level = 0.95,
  option = "SM",
  stdz = "indirect",
  alternative = "two.sided",
  ...
)
```

Arguments

object	a model fitted from linear_cre.
parm	specify a subset of providers for which confidence intervals are given. By default, all providers are included. The class of parm should match the class of the provider IDs.
level	the confidence level. The default value is 0.95.
option	a character string specifying whether the confidence intervals should be provided for provider effects or standardized measures: • "alpha" provider effect. • "SM" standardized measures.
stdz	a character string or a vector specifying the standardization method if option includes "SM". See stdz argument in SM_output.linear_re.
alternative	a character string specifying the alternative hypothesis, must be one of "two.sided" (default), "greater", or "less". Note that "alpha" for argument option only supports "two.sided".
...	additional arguments that can be passed to the function.

Value

A list of data frames containing the confidence intervals based on the values of option and stdz.

CI.alpha	Confidence intervals for provider effects if option includes "alpha".
CI.indirect	Confidence intervals for indirect standardized differences if option includes "SM" and stdz includes "indirect".
CI.direct	Confidence intervals for direct standardized differences if option includes "SM" and stdz includes "direct".

Examples

```
data(ExampleDataLinear)
outcome <- ExampleDataLinear$Y
covar <- ExampleDataLinear$Z
ProvID <- ExampleDataLinear$ProvID
data <- data.frame(outcome, ProvID, covar)
outcome.char <- colnames(data)[1]
ProvID.char <- colnames(data)[2]
wb.char <- c("z1", "z2")
other.char <- c("z3", "z4", "z5")
fit_cre <- linear_cre(data = data, Y.char = outcome.char, ProvID.char = ProvID.char,
wb.char = wb.char, other.char = other.char)
confint(fit_cre)
```

`confint.linear_fe` *Get confidence intervals for provider effects or standardized measures from a fitted* `linear_fe` *object*

Description

Provide confidence intervals for provider effects or standardized measures from from a fixed effect linear model.

Usage

```
## S3 method for class 'linear_fe'
confint(
  object,
  parm,
  level = 0.95,
  option = "SM",
  stdz = "indirect",
  null = "median",
  alternative = "two.sided",
  ...
)
```

Arguments

`object`	a model fitted from `linear_fe`.
`parm`	specify a subset of providers for which confidence intervals are given. By default, all providers are included. The class of `parm` should match the class of the provider IDs.
`level`	the confidence level. The default value is 0.95.
`option`	a character string specifying whether the confidence intervals should be provided for provider effects or standardized measures: • `"gamma"` provider effect (only supports `"two.sided"` confidence interval). • `"SM"` standardized measures.
`stdz`	a character string or a vector specifying the standardization method if `option` includes `"SM"`. See `stdz` argument in `SM_output.linear_fe`.
`null`	a character string or a number specifying the population norm for calculating standardized measures if `option` includes `"SM"`. See `null` argument in `SM_output.linear_fe`.
`alternative`	a character string specifying the alternative hypothesis, must be one of `"two.sided"` (default), `"greater"`, or `"less"`. Note that `"gamma"` for argument `option` only supports `"two.sided"`.
`...`	additional arguments that can be passed to the function.

Value

A list of data frames containing the confidence intervals based on the values of `option` and `stdz`.

`CI.gamma`	Confidence intervals for provider effects if `option` includes `"gamma"`.
`CI.indirect`	Confidence intervals for indirect standardized differences if `option` includes `"SM"` and `stdz` includes `"indirect"`.
`CI.direct`	Confidence intervals for direct standardized differences if `option` includes `"SM"` and `stdz` includes `"direct"`.

Examples

```
data(ExampleDataLinear)
outcome <- ExampleDataLinear$Y
covar <- ExampleDataLinear$Z
ProvID <- ExampleDataLinear$ProvID
fit_linear <- linear_fe(Y = outcome, Z = covar, ProvID = ProvID)
confint(fit_linear)
```

`confint.linear_re` *Get confidence intervals for provider effects or standardized measures from a fitted* `linear_re` *object*

Description

Provide confidence intervals for provider effects or standardized measures from a random effect linear model.

Usage

```
## S3 method for class 'linear_re'
confint(
  object,
  parm,
  level = 0.95,
  option = "SM",
  stdz = "indirect",
  alternative = "two.sided",
  ...
)
```

Arguments

`object`	a model fitted from `linear_re`.
`parm`	specify a subset of providers for which confidence intervals are given. By default, all providers are included. The class of `parm` should match the class of the provider IDs.

level	the confidence level. The default value is 0.95.
option	a character string specifying whether the confidence intervals should be provided for provider effects or standardized measures: • "alpha" provider effect. • "SM" standardized measures.
stdz	a character string or a vector specifying the standardization method if option includes "SM". See stdz argument in SM_output.linear_re.
alternative	a character string specifying the alternative hypothesis, must be one of "two.sided" (default), "greater", or "less". Note that "alpha" for argument option only supports "two.sided".
...	additional arguments that can be passed to the function.

Value

A list of data frames containing the confidence intervals based on the values of option and stdz.

CI.alpha	Confidence intervals for provider effects if option includes "alpha".
CI.indirect	Confidence intervals for indirect standardized differences if option includes "SM" and stdz includes "indirect".
CI.direct	Confidence intervals for direct standardized differences if option includes "SM" and stdz includes "direct".

Examples

```
data(ExampleDataLinear)
outcome <- ExampleDataLinear$Y
ProvID <- ExampleDataLinear$ProvID
covar <- ExampleDataLinear$Z
fit_re <- linear_re(Y = outcome, Z = covar, ProvID = ProvID)
confint(fit_re)
```

---

confint.logis_cre *Get confidence intervals for provider effects or standardized measures from a fitted* logis_cre *object*

Description

Provide confidence intervals for provider effects or standardized measures from a correlated random effect logistic model.

Usage

```
## S3 method for class 'logis_cre'
confint(
  object,
  parm,
  level = 0.95,
  option = "SM",
  measure = c("rate", "ratio"),
  stdz = "indirect",
  alternative = "two.sided",
  ...
)
```

Arguments

object	a model fitted from logis_cre.
parm	specify a subset of providers for which confidence intervals are given. By default, all providers are included. The class of parm should match the class of the provider IDs.
level	the confidence level. The default value is 0.95.
option	a character string specifying whether the confidence intervals should be provided for provider effects or standardized measures: • "alpha" provider effect. • "SM" standardized measures.
measure	a character string or a vector indicating whether the output measure is "ratio" or "rate" if option = "SM". Both "rate" and "ratio" will be provided by default.
stdz	a character string or a vector specifying the standardization method if option includes "SM". See stdz argument in SM_output.linear_re.
alternative	a character string specifying the alternative hypothesis, must be one of "two.sided" (default), "greater", or "less". Note that "alpha" for argument option only supports "two.sided".
...	additional arguments that can be passed to the function.

Value

A list of data frames containing the confidence intervals based on the values of option and stdz.

CI.alpha	Confidence intervals for provider effects if option includes "alpha".
CI.indirect	Confidence intervals for indirect standardized differences if option includes "SM" and stdz includes "indirect".
CI.direct	Confidence intervals for direct standardized differences if option includes "SM" and stdz includes "direct".

Examples

```
data(ExampleDataBinary)
outcome <- ExampleDataBinary$Y
covar <- ExampleDataBinary$Z
ProvID <- ExampleDataBinary$ProvID
data <- data.frame(outcome, ProvID, covar)
outcome.char <- colnames(data)[1]
ProvID.char <- colnames(data)[2]
wb.char <- c("z1", "z2")
other.char <- c("z3", "z4", "z5")
fit_cre <- logis_cre(data = data, Y.char = outcome.char, ProvID.char = ProvID.char,
wb.char = wb.char, other.char = other.char)
confint(fit_cre)
```

`confint.logis_fe` *Get confidence intervals for provider effects or standardized measures from a fitted* `logis_fe` *object*

Description

Provide confidence intervals for provider effects or standardized measures from a fixed effect logistic model.

Usage

```
## S3 method for class 'logis_fe'
confint(
  object,
  parm,
  level = 0.95,
  test = "exact",
  option = "SM",
  stdz = "indirect",
  null = "median",
  measure = c("rate", "ratio"),
  alternative = "two.sided",
  ...
)
```

Arguments

`object`	a model fitted from `logis_fe`.
`parm`	specify a subset of providers for which confidence intervals are given. By default, all providers are included. The class of `parm` should match the class of the provider IDs.
`level`	the confidence level. The default value is 0.95.

`test` a character string specifying the type of testing method. The default is "exact".

- `"exact"` exact test.
- `"wald"` wald test.
- `"score"` score test.

`option` a character string specifying whether the confidence intervals should be provided for provider effects or standardized measures:

- `"gamma"` provider effect.
- `"SM"` standardized measures.

`stdz` a character string or a vector specifying the standardization method if `option = "SM"`. See `stdz` argument in `SM_output.logis_fe`.

`null` a character string or a number defining the population norm if `option = "SM"`.

`measure` a character string or a vector indicating whether the output measure is "ratio" or "rate" if `option = "SM"`. Both "rate" and "ratio" will be provided by default.

- `"rate"` output the standardized rate. The "rate" has been restricted to 0% - 100%.
- `"ratio"` output the standardized ratio.
- `c("ratio", "rate")` output both the standardized rate and ratio.

`alternative` a character string specifying the alternative hypothesis, must be one of `"two.sided"` (default), `"greater"`, or `"less"`. Note that `"gamma"` for argument `option` only supports `"two.sided"`.

`...` additional arguments that can be passed to the function.

Details

The wald test is invalid for extreme providers (i.e. when provider effect goes to infinity). We suggest using score or exact test to generate confidence intervals.

Value

A dataframe (`option = "gamma"`) or a list of data frames (`option = "SM"`) containing the point estimate, and lower and upper bounds of the estimate.

Examples

```
data(ExampleDataBinary)
outcome = ExampleDataBinary$Y
covar = ExampleDataBinary$Z
ProvID = ExampleDataBinary$ProvID
fit_fe <- logis_fe(Y = outcome, Z = covar, ProvID = ProvID, message = FALSE)
confint(fit_fe, option = "gamma")
confint(fit_fe, option = "SM")
```

`confint.logis_re` *Get confidence intervals for provider effects or standardized measures from a fitted* `logis_re` *object*

Description

Provide confidence intervals for provider effects or standardized measures from a random effect logistic model.

Usage

```
## S3 method for class 'logis_re'
confint(
  object,
  parm,
  level = 0.95,
  option = "SM",
  measure = c("rate", "ratio"),
  stdz = "indirect",
  alternative = "two.sided",
  ...
)
```

Arguments

`object`	a model fitted from `logis_re`.
`parm`	specify a subset of providers for which confidence intervals are given. By default, all providers are included. The class of `parm` should match the class of the provider IDs.
`level`	the confidence level. The default value is 0.95.
`option`	a character string specifying whether the confidence intervals should be provided for provider effects or standardized measures: • `"alpha"` provider effect. • `"SM"` standardized measures.
`measure`	a character string or a vector indicating whether the output measure is "ratio" or "rate" if `option = "SM"`. Both "rate" and "ratio" will be provided by default.
`stdz`	a character string or a vector specifying the standardization method if `option` includes `"SM"`. See `stdz` argument in `SM_output.linear_re`.
`alternative`	a character string specifying the alternative hypothesis, must be one of `"two.sided"` (default), `"greater"`, or `"less"`. Note that `"alpha"` for argument `option` only supports `"two.sided"`.
`...`	additional arguments that can be passed to the function.

Value

A list of data frames containing the confidence intervals based on the values of `option` and `stdz`.

`CI.alpha`	Confidence intervals for provider effects if `option` includes `"alpha"`.
`CI.indirect`	Confidence intervals for indirect standardized differences if `option` includes `"SM"` and `stdz` includes `"indirect"`.
`CI.direct`	Confidence intervals for direct standardized differences if `option` includes `"SM"` and `stdz` includes `"direct"`.

Examples

```
data(ExampleDataBinary)
outcome <- ExampleDataBinary$Y
ProvID <- ExampleDataBinary$ProvID
covar <- ExampleDataBinary$Z
fit_re <- logis_re(Y = outcome, Z = covar, ProvID = ProvID)
confint(fit_re)
```

---

data_check *Data quality check function*

Description

Conduct data quality check including checking missingness, variation, correlation and VIF of variables.

Usage

```
data_check(Y, Z, ProvID)
```

Arguments

`Y`	a numeric vector indicating the outcome variable.
`Z`	a matrix or data frame representing covariates.
`ProvID`	a numeric vector representing the provider identifier.

Details

The function performs the following checks:

- **Missingness:** Checks for any missing values in the dataset and provides a summary of missing data.
- **Variation:** Identifies covariates with zero or near-zero variance which might affect model stability.
- **Correlation:** Analyzes pairwise correlation among covariates and highlights highly correlated pairs.

- **VIF:** Computes the Variable Inflation Factors to identify covariates with potential multicollinearity issues.

If issues arise when using the model functions `logis_fe`, `linear_fe` and `linear_re`, this function can be called for data quality checking purposes.

Value

No return value, called for side effects.

Examples

```
data(ExampleDataBinary)
outcome = ExampleDataBinary$Y
covar = ExampleDataBinary$Z
ProvID = ExampleDataBinary$ProvID
data_check(outcome, covar, ProvID)
```

---

ecls_data *Early Childhood Longitudinal Study Dataset*

Description

The Early Childhood Longitudinal Study (ECLS) dataset tracks more than 18,000 children from kindergarten through fifth grade, providing comprehensive student-level information. See Tourangeau et al. (2015) for details.

Usage

```
data(ecls_data)
```

Format

A data frame with 9,101 observations, including:

Child_ID Unique identifier for each child in the dataset.

School_ID Identifier for each school in the dataset.

Math_Score Continuous variable representing the consolidated math proficiency score.

Income Household income, categorized into 18 ordinal levels, from $5,000 or less to $200,000 or more. Treated as a continuous variable in the analysis.

Child_Sex Binary variable indicating the gender of each student.

Details

The dataset includes fifth-grade cross-sectional data, focusing on students' mathematical assessment scores as the primary outcome measure. The mathematical assessment covers 18 topics such as number sense, properties, operations, measurement, geometry, data analysis, and algebra. These items evaluate students' competencies in conceptual knowledge, procedural knowledge, and problem-solving, consolidated into a single "Math score." A higher score indicates greater proficiency. The primary predictors are household income and gender, where gender is a categorical variable. Household income is categorized into 18 ordinal levels, ranging from $5,000 or less (level 1) to $200,000 or more (level 18). In the analysis, the income variable is treated as a continuous variable. The dataset removes all records with missing values and consists of 9,101 complete observations from 2,275 schools.

Source

Available at the following website: https://nces.ed.gov/ecls/.

References

Tourangeau, K., Nord, C., Lê, T., Sorongon, A. G., Hagedorn, M. C., Daly, P., & Najarian, M. (2015). Early Childhood Longitudinal Study, Kindergarten Class of 2010-11 (ECLS-K:2011): User's manual for the ECLS-K:2011 kindergarten data file and electronic codebook, public version (NCES 2015-074). National Center for Education Statistics.

Examples

```
data(ecls_data)
formula_FE <- as.formula("Math_Score ~ Income + id(School_ID) + Child_Sex")
fit_FE <- linear_fe(formula = formula_FE, data = ecls_data)

formula_RE <- as.formula("Math_Score ~ Income + (1|School_ID) + Child_Sex")
fit_RE <- linear_re(formula = formula_RE, data = ecls_data)
```

ExampleDataBinary *Example data with binary outcomes*

Description

A simulated data set containing 7994 observations, 5 continuous covariates and 100 providers.

Usage

```
data(ExampleDataBinary)
```

Format

A list containing the following elements:

Y a vector representing binary outcomes with 0 or 1. Generated from a Bernoulli distribution with the probability of success ($\boldsymbol{\mu}$), which is determined by applying the logistic function to the linear combination of provider effects and covariates.

ProvID a vector representing identifier for each provider. The number of individuals per provider is generated from a Poisson distribution with mean 80, with a minimum value of 11.

Z a data frame containing 5 continuous variables. Generated from a multivariate normal distribution, where the mean is calculated as $(\gamma_i - \mu_\gamma) \cdot \rho/\sigma_\gamma$ with $\mu_\gamma = \log(4/11)$ and $\sigma_\gamma = 0.4$, and the correlation between covariates and provider effects is 0.1.

Examples

```
data(ExampleDataBinary)
head(ExampleDataBinary$Y)
head(ExampleDataBinary$ProvID)
head(ExampleDataBinary$Z)
```

`ExampleDataLinear` *Example data with continuous outcomes*

Description

A simulated data set containing 7901 observations, 5 continuous covariates and 100 providers.

Usage

```
data(ExampleDataLinear)
```

Format

A list containing the following elements:

Y a vector representing continuous outcomes. Generated as a linear combination of provider-specific effects, covariates, and a random error term (ϵ), where ϵ follows a normal distribution with mean 0 and standard deviation 1.

ProvID a vector representing identifier for each provider. The number of individuals per provider is generated from a Poisson distribution with mean 80.

Z a data frame containing 5 continuous variables. Generated from a multivariate normal distribution, where the mean is calculated as $(\gamma_i - \mu_\gamma) \cdot \rho/\sigma_\gamma$ with $\mu_\gamma = \log(4/11)$ and $\sigma_\gamma = 0.4$, and the correlation between covariates and provider effects is 0.1.

Examples

```
data(ExampleDataLinear)
head(ExampleDataLinear$Y)
head(ExampleDataLinear$ProvID)
head(ExampleDataLinear$Z)
```

`linear_cre` *Main Function for fitting correlated random effect linear model*

Description

Fit a correlated random effect linear model via `lmer` from the `lme4` package.

Usage

```
linear_cre(data, Y.char, wb.char, other.char = NULL, ProvID.char, ...)
```

Arguments

`data`	a data frame containing all variables.
`Y.char`	a character string specifying the column name of the response variable in the `data`.
`wb.char`	a character vector specifying covariates to be decomposed into within (`*_within`) and between (`*_bar`) components.
`other.char`	a character vector specifying additional covariates to include in the model without decomposition.
`ProvID.char`	a character string specifying the column name of the provider identifier in the `data`.
`...`	additional arguments passed to `lmer` for further customization.

Details

Fit a correlated random effect linear model using `lmer` with a Mundlak within–between decomposition for selected covariates. For each decomposed covariate Z_k, the function constructs $Z_{k,\mathrm{bar},i} = \frac{1}{n_i}\sum_j Z_{k,ij}$ (the group mean, "between") and $Z_{k,\mathrm{within},ij} = Z_{k,ij} - Z_{k,\mathrm{bar},i}$ (the within-group deviation), and estimates

$$Y_{ij} = \mu + \alpha_i + \sum_k \beta_{k,W} Z_{k,\mathrm{within},ij} + \sum_k \beta_{k,B} Z_{k,\mathrm{bar},i} + \mathbf{X}_{ij}^{\top}\gamma + \varepsilon_{ij},$$

where $\alpha_i \sim \mathcal{N}(0, \sigma_\alpha^2)$ is a random intercept.

The function creates, for every name in `wb.char`, two columns: `<var>_bar` (group mean within `ProvID.char`) and `<var>_within` (observation minus its group mean). The fitted model is:

```
Y ~ <all *_within> + <all *_bar> + <other.char> + (1 | ProvID)
```

In addition to these input formats, all arguments from the `lmer` function can be modified via `...`, allowing for customization of model fitting options such as controlling the optimization method or adjusting convergence criteria. By default, the model is fitted using REML (restricted maximum likelihood).

If issues arise during model fitting, consider using the `data_check` function to perform a data quality check, which can help identify missing values, low variation in covariates, high-pairwise correlation, and multicollinearity. For datasets with missing values, this function automatically removes observations (rows) with any missing values before fitting the model.

Value

A list of objects with S3 class "linear_cre":

coefficient	a list containing the estimated coefficients: FE, the fixed effects for each predictor and the intercept, and RE, the random effects for each provider.
variance	a list containing the variance estimates: FE, the variance-covariance matrix of the fixed effect coefficients, and RE, the variance of the random effects.
sigma	the residual standard error.
fitted	the fitted values of each individual.
observation	the original response of each individual.
residuals	the residuals of each individual, that is response minus fitted values.
linear_pred	the linear predictor of each individual.
data_include	the processed data used to fit the model, sorted by the provider identifier. This includes the within-group (*_within) and between-group (*_bar) components for variables specified in wb.char. For categorical covariates, it includes the dummy variables created for all categories except the reference level.
char_list	a list of the character vectors representing the column names for the response variable, covariates, and provider identifier. For categorical variables, the names reflect the dummy variables created for each category.
Loglkd	the log-likelihood.
AIC	Akaike information criterion.
BIC	Bayesian information criterion.

References

Bates D, Maechler M, Bolker B, Walker S (2015). *Fitting Linear Mixed-Effects Models Using lme4*. Journal of Statistical Software, 67(1), 1-48.

See Also

data_check

Examples

```
data(ExampleDataLinear)
outcome <- ExampleDataLinear$Y
covar <- ExampleDataLinear$Z
ProvID <- ExampleDataLinear$ProvID
data <- data.frame(outcome, ProvID, covar)
outcome.char <- colnames(data)[1]
ProvID.char <- colnames(data)[2]
wb.char <- c("z1", "z2")
other.char <- c("z3", "z4", "z5")

# Fit a correlated random effect linear model
```

```
fit_cre <- linear_cre(data = data, Y.char = outcome.char, ProvID.char = ProvID.char,
wb.char = wb.char, other.char = other.char)
```

`linear_fe` *Main function for fitting the fixed effect linear model*

Description

Fit a fixed effect linear model via profile likelihood.

Usage

```
linear_fe(
  formula = NULL,
  data = NULL,
  Y = NULL,
  Z = NULL,
  ProvID = NULL,
  Y.char = NULL,
  Z.char = NULL,
  ProvID.char = NULL,
  option.gamma.var = "simplified"
)
```

Arguments

`formula`	a two-sided formula object describing the model to be fitted, with the response variable on the left of a ~ operator and covariates on the right, separated by + operators. The fixed effect of the provider identifier is specified using `id()`.
`data`	a data frame containing the variables named in the `formula`, or the columns specified by `Y.char`, `Z.char`, and `ProvID.char`.
`Y`	a numeric vector representing the response variable.
`Z`	a matrix or data frame representing the covariates, which can include both numeric and categorical variables.
`ProvID`	a numeric vector representing the provider identifier.
`Y.char`	a character string specifying the column name of the response variable in the `data`.
`Z.char`	a character vector specifying the column names of the covariates in the `data`.
`ProvID.char`	a character string specifying the column name of the provider identifier in the `data`.
`option.gamma.var`	a character string specifying the method to calculate the variance of provider effects gamma, must be `"full"` or `"simplified"`. You can specify just the initial letter.

- `"simplified"` (default) calculating the simplified variance of provider effects assuming regression coefficients are known. This approach is suitable for large datasets where the results of the full and simplified methods are similar, or when the full method may become unstable due to complex settings.
- `"full"` considering the correlation between provider effects and regression coefficients.

Details

This function is used to fit a fixed effect linear model of the form:

$$Y_{ij} = \gamma_i + \mathbf{Z}_{ij}^{\top}\boldsymbol{\beta} + \epsilon_{ij}$$

where Y_{ij} is the continuous outcome for individual j in provider i, γ_i is the provider-specific effect, $\mathbf{Z}_{ij}$ are the covariates, and $\boldsymbol{\beta}$ is the vector of coefficients for the covariates.

The function accepts three different input formats: a formula and dataset, where the formula is of the form `response ~ covariates + id(provider)`, with `provider` representing the provider identifier; a dataset along with the column names of the response, covariates, and provider identifier; or the outcome vector $\boldsymbol{Y}$, the covariate matrix or data frame $\mathbf{Z}$, and the provider identifier vector.

If issues arise during model fitting, consider using the `data_check` function to perform a data quality check, which can help identify missing values, low variation in covariates, high-pairwise correlation, and multicollinearity. For datasets with missing values, this function automatically removes observations (rows) with any missing values before fitting the model.

Value

A list of objects with S3 class `"linear_fe"`:

`coefficient`	a list containing the estimated coefficients: `beta`, the fixed effects for each predictor, and `gamma`, the effect for each provider.
`variance`	a list containing the variance estimates: `beta`, the variance-covariance matrix of the predictor coefficients, and `gamma`, the variance of the provider effects.
`sigma`	the residual standard error.
`fitted`	the fitted values of each individual.
`observation`	the original response of each individual.
`residuals`	the residuals of each individual, that is response minus fitted values.
`linear_pred`	the linear predictor of each individual.
`data_include`	the data used to fit the model, sorted by the provider identifier. For categorical covariates, this includes the dummy variables created for all categories except the reference level.
`char_list`	a list of the character vectors representing the column names for the response variable, covariates, and provider identifier. For categorical variables, the names reflect the dummy variables created for each category.
`Loglkd`	log likelihood.
`AIC`	Akaike information criterion.
`BIC`	Bayesian information criterion.

References

Hsiao, C. (2022). Analysis of panel data (No. 64). Cambridge university press.

See Also

data_check

Examples

```
data(ExampleDataLinear)
outcome <- ExampleDataLinear$Y
covar <- ExampleDataLinear$Z
ProvID <- ExampleDataLinear$ProvID
data <- data.frame(outcome, ProvID, covar)
covar.char <- colnames(covar)
outcome.char <- colnames(data)[1]
ProvID.char <- colnames(data)[2]
formula <- as.formula(paste("outcome ~", paste(covar.char, collapse = " + "), "+ id(ProvID)"))

# Fit fixed linear effect model using three input formats
fit_fe1 <- linear_fe(Y = outcome, Z = covar, ProvID = ProvID)
fit_fe2 <- linear_fe(data = data, Y.char = outcome.char,
Z.char = covar.char, ProvID.char = ProvID.char)
fit_fe3 <- linear_fe(formula, data)
```

---

linear_re *Main Function for fitting the random effect linear model*

Description

Fit a random effect linear model via lmer from the lme4 package.

Usage

```
linear_re(
  formula = NULL,
  data = NULL,
  Y = NULL,
  Z = NULL,
  ProvID = NULL,
  Y.char = NULL,
  Z.char = NULL,
  ProvID.char = NULL,
  ...
)
```

Arguments

formula	a two-sided formula object describing the model to be fitted, with the response variable on the left of a ~ operator and covariates on the right, separated by + operators. The random effect of the provider identifier is specified using (1 |).
data	a data frame containing the variables named in the formula, or the columns specified by Y.char, Z.char, and ProvID.char.
Y	a numeric vector representing the response variable.
Z	a matrix or data frame representing the covariates, which can include both numeric and categorical variables.
ProvID	a numeric vector representing the provider identifier.
Y.char	a character string specifying the column name of the response variable in the data.
Z.char	a character vector specifying the column names of the covariates in the data.
ProvID.char	a character string specifying the column name of the provider identifier in the data.
...	additional arguments passed to lmer for further customization.

Details

This function is used to fit a random effect linear model of the form:

$$Y_{ij} = \mu + \alpha_i + \mathbf{Z}_{ij}^{\top}\boldsymbol{\beta} + \epsilon_{ij}$$

where Y_{ij} is the continuous outcome for individual j in provider i, μ is the overall intercept, α_i is the random effect for provider i, $\mathbf{Z}_{ij}$ are the covariates, and $\boldsymbol{\beta}$ is the vector of coefficients for the covariates.

The model is fitted by overloading the lmer function from the lme4 package. Three different input formats are accepted: a formula and dataset, where the formula is of the form response ~ covariates + (1 | provider), with provider representing the provider identifier; a dataset along with the column names of the response, covariates, and provider identifier; or the outcome vector $\boldsymbol{Y}$, the covariate matrix or data frame $\mathbf{Z}$, and the provider identifier vector.

In addition to these input formats, all arguments from the lmer function can be modified via ..., allowing for customization of model fitting options such as controlling the optimization method or adjusting convergence criteria. By default, the model is fitted using REML (restricted maximum likelihood).

If issues arise during model fitting, consider using the data_check function to perform a data quality check, which can help identify missing values, low variation in covariates, high-pairwise correlation, and multicollinearity. For datasets with missing values, this function automatically removes observations (rows) with any missing values before fitting the model.

Value

A list of objects with S3 class "linear_re":

coefficient	a list containing the estimated coefficients: FE, the fixed effects for each predictor and the intercept, and RE, the random effects for each provider.

variance	a list containing the variance estimates: FE, the variance-covariance matrix of the fixed effect coefficients, and RE, the variance of the random effects.
sigma	the residual standard error.
fitted	the fitted values of each individual.
observation	the original response of each individual.
residuals	the residuals of each individual, that is response minus fitted values.
linear_pred	the linear predictor of each individual.
data_include	the data used to fit the model, sorted by the provider identifier. For categorical covariates, this includes the dummy variables created for all categories except the reference level.
char_list	a list of the character vectors representing the column names for the response variable, covariates, and provider identifier. For categorical variables, the names reflect the dummy variables created for each category.
Loglkd	the log-likelihood.
AIC	Akaike information criterion.
BIC	Bayesian information criterion.

References

Bates D, Maechler M, Bolker B, Walker S (2015). *Fitting Linear Mixed-Effects Models Using lme4*. Journal of Statistical Software, 67(1), 1-48.

See Also

data_check

Examples

```
data(ExampleDataLinear)
outcome <- ExampleDataLinear$Y
covar <- ExampleDataLinear$Z
ProvID <- ExampleDataLinear$ProvID
data <- data.frame(outcome, ProvID, covar)
covar.char <- colnames(covar)
outcome.char <- colnames(data)[1]
ProvID.char <- colnames(data)[2]
formula <- as.formula(paste("outcome ~", paste(covar.char, collapse = " + "), "+ (1|ProvID)"))

# Fit random effect linear model using three input formats
fit_re1 <- linear_re(Y = outcome, Z = covar, ProvID = ProvID)
fit_re2 <- linear_re(data = data, Y.char = outcome.char,
Z.char = covar.char, ProvID.char = ProvID.char)
fit_re3 <- linear_re(formula, data)
```

`logis_cre` *Main Function for fitting correlated random effect logistic model*

Description

Fit a correlated random effect logistic model via `glmer` from the `lme4` package.

Usage

```
logis_cre(data, Y.char, wb.char, other.char = NULL, ProvID.char, ...)
```

Arguments

`data`	a data frame containing all variables.
`Y.char`	a character string specifying the column name of the response variable in the `data`.
`wb.char`	a character vector specifying covariates to be decomposed into within (`*_within`) and between (`*_bar`) components.
`other.char`	a character vector specifying additional covariates to include in the model without decomposition.
`ProvID.char`	a character string specifying the column name of the provider identifier in the `data`.
`...`	additional arguments passed to `glmer` for further customization.

Details

Fit a correlated random effect logistic model using `glmer` with a Mundlak within–between decomposition for selected covariates. For each decomposed covariate Z_k, the function constructs $Z_{k,\text{bar},i} = \frac{1}{n_i}\sum_j Z_{k,ij}$ (the group mean, "between") and $Z_{k,\text{within},ij} = Z_{k,ij} - Z_{k,\text{bar},i}$ (the within-group deviation), and estimates the model

$$\text{logit}\left(P(Y_{ij}=1)\right) = \mu + \alpha_i + \sum_k \beta_{k,W} Z_{k,\text{within},ij} + \sum_k \beta_{k,B} Z_{k,\text{bar},i} + \mathbf{X}_{ij}^{\top}\gamma,$$

where $\alpha_i \sim \mathcal{N}(0, \sigma_\alpha^2)$ is a random intercept.

The function creates, for every name in `wb.char`, two columns: `<var>_bar` (group mean within `ProvID.char`) and `<var>_within` (observation minus its group mean). The fitted model formula is:

```
Y ~ <all *_within> + <all *_bar> + <other.char> + (1 | ProvID)
```

This model is fitted using `glmer` with `family = binomial(link = "logit")`.

In addition to these input formats, all arguments from the `glmer` function can be modified via `...`, allowing for customization of model fitting options such as controlling the optimization method or adjusting convergence criteria.

If issues arise during model fitting, consider using the `data_check` function to perform a data quality check. For datasets with missing values, this function automatically removes observations (rows) with any missing values before fitting the model.

Value

A list of objects with S3 class "logis_cre":

coefficient	a list containing the estimated coefficients: FE, the fixed effects for each predictor and the intercept, and RE, the random effects for each provider.
variance	a list containing the variance estimates: FE, the variance-covariance matrix of the fixed effect coefficients, and RE, the variance of the random effects.
fitted	the fitted values of each individual.
observation	the original response of each individual.
linear_pred	the linear predictor of each individual.
data_include	the processed data used to fit the model, sorted by the provider identifier. This includes the within-group (*_within) and between-group (*_bar) components for variables specified in wb.char. For categorical covariates, it includes the dummy variables created for all categories except the reference level.
char_list	a list of the character vectors representing the column names for the response variable, provider identifier, the generated names for covariates with decomposition (wb.char, and covariates included without decomposition (other.char). For categorical variables, the names reflect the dummy variables created for each category.
Loglkd	the log-likelihood.
AIC	Akaike information criterion.
BIC	Bayesian information criterion.

References

Bates D, Maechler M, Bolker B, Walker S (2015). *Fitting Linear Mixed-Effects Models Using lme4*. Journal of Statistical Software, 67(1), 1-48.

See Also

data_check

Examples

```
data(ExampleDataBinary)
outcome <- ExampleDataBinary$Y
covar <- ExampleDataBinary$Z
ProvID <- ExampleDataBinary$ProvID
data <- data.frame(outcome, ProvID, covar)
outcome.char <- colnames(data)[1]
ProvID.char <- colnames(data)[2]
wb.char <- c("z1", "z2")
other.char <- c("z3", "z4", "z5")

# Fit a correlated random effect linear model
fit_cre <- logis_cre(data = data, Y.char = outcome.char, ProvID.char = ProvID.char,
```

```
wb.char = wb.char, other.char = other.char)
```

`logis_fe` *Main function for fitting the fixed effect logistic model*

Description

Fit a fixed effect logistic model via Serial blockwise inversion Newton (SerBIN) or block ascent Newton (BAN) algorithm.

Usage

```
logis_fe(
  formula = NULL,
  data = NULL,
  Y.char = NULL,
  Z.char = NULL,
  ProvID.char = NULL,
  Y = NULL,
  Z = NULL,
  ProvID = NULL,
  method = "SerBIN",
  max.iter = 10000,
  tol = 1e-05,
  bound = 10,
  cutoff = 10,
  backtrack = TRUE,
  stop = "or",
  threads = 1,
  message = TRUE
)
```

Arguments

`formula`	a two-sided formula object describing the model to be fitted, with the response variable on the left of a ~ operator and covariates on the right, separated by + operators. The fixed effect of the provider identifier is specified using `id()`.
`data`	a data frame containing the variables named in the `formula`, or the columns specified by `Y.char`, `Z.char`, and `ProvID.char`.
`Y.char`	a character string specifying the column name of the response variable in the `data`.
`Z.char`	a character vector specifying the column names of the covariates in the `data`.
`ProvID.char`	a character string specifying the column name of the provider identifier in the `data`.
`Y`	a numeric vector representing the response variable.

`Z` a matrix or data frame representing the covariates, which can include both numeric and categorical variables.

`ProvID` a numeric vector representing the provider identifier.

`method` a string specifying the algorithm to be used. The default value is "SerBIN".

- `"SerBIN"` uses the Serial blockwise inversion Newton algorithm to fit the model (See Wu et al. (2022)).
- `"BAN"` uses the block ascent Newton algorithm to fit the model (See He et al. (2013)).

`max.iter` maximum iteration number if the stopping criterion specified by `stop` is not satisfied. The default value is 10,000.

`tol` tolerance used for stopping the algorithm. See details in `stop` below. The default value is 1e-5.

`bound` a positive number to avoid inflation of provider effects. The default value is 10.

`cutoff` An integer specifying the minimum number of observations required for providers. Providers with fewer observations than the cutoff will be labeled as `"include = 0"` and excluded from model fitting. The default is 10.

`backtrack` a Boolean indicating whether backtracking line search is implemented. The default is FALSE.

`stop` a character string specifying the stopping rule to determine convergence.

- `"beta"` stop the algorithm when the infinity norm of the difference between current and previous beta coefficients is less than the `tol`.
- `"relch"` stop the algorithm when the $(loglik(m) - loglik(m-1))/(loglik(m))$ (the difference between the log-likelihood of the current iteration and the previous iteration divided by the log-likelihood of the current iteration) is less than the `tol`.
- `"ratch"` stop the algorithm when $(loglik(m) - loglik(m-1))/(loglik(m) - loglik(0))$ (the difference between the log-likelihood of the current iteration and the previous iteration divided by the difference of the log-likelihood of the current iteration and the initial iteration) is less than the `tol`.
- `"all"` stop the algorithm when all the stopping rules (`"beta"`, `"relch"`, `"ratch"`) are met.
- `"or"` stop the algorithm if any one of the rules (`"beta"`, `"relch"`, `"ratch"`) is met.

The default value is `or`. If `iter.max` is achieved, it overrides any stop rule for algorithm termination.

`threads` a positive integer specifying the number of threads to be used. The default value is 1.

`message` a Boolean indicating whether to print the progress of the fitting process. The default is TRUE.

Details

The function accepts three different input formats: a formula and dataset, where the formula is of the form `response ~ covariates + id(provider)`, with `provider` representing the provider

identifier; a dataset along with the column names of the response, covariates, and provider identifier; or the binary outcome vector $\boldsymbol{Y}$, the covariate matrix or data frame $\mathbf{Z}$, and the provider identifier vector.

The default algorithm is based on Serial blockwise inversion Newton (SerBIN) proposed by Wu et al. (2022), but users can also choose to use the block ascent Newton (BAN) algorithm proposed by He et al. (2013) to fit the model. Both methodologies build upon the Newton-Raphson method, yet SerBIN simultaneously updates both the provider effect and covariate coefficient. This concurrent update necessitates the inversion of the whole information matrix at each iteration. In contrast, BAN adopts a two-layer updating approach, where the covariate coefficient is sequentially fixed to update the provider effect, followed by fixing the provider effect to update the covariate coefficient.

We suggest using the default `"SerBIN"` option as it typically converges subsequently much faster for most datasets. However, in rare cases where the SerBIN algorithm encounters second-order derivative irreversibility leading to an error, users can consider using the `"BAN"` option as an alternative. For a deeper understanding, please consult the original article for detailed insights.

If issues arise during model fitting, consider using the `data_check` function to perform a data quality check, which can help identify missing values, low variation in covariates, high-pairwise correlation, and multicollinearity. For datasets with missing values, this function automatically removes observations (rows) with any missing values before fitting the model.

Value

A list of objects with S3 class `"logis_fe"`:

`coefficient`	a list containing the estimated coefficients: `beta`, the fixed effects for each predictor, and `gamma`, the effect for each provider.
`variance`	a list containing the variance estimates: `beta`, the variance-covariance matrix of the predictor coefficients, and `gamma`, the variance of the provider effects.
`linear_pred`	the linear predictor of each individual.
`fitted`	the predicted probability of each observation having a response of 1.
`observation`	the original response of each individual.
`Loglkd`	the log-likelihood.
`AIC`	Akaike info criterion.
`BIC`	Bayesian info criterion.
`AUC`	area under the ROC curve.
`char_list`	a list of the character vectors representing the column names for the response variable, covariates, and provider identifier. For categorical variables, the names reflect the dummy variables created for each category.
`data_include`	the data used to fit the model, sorted by the provider identifier. For categorical covariates, this includes the dummy variables created for all categories except the reference level. Additionally, it contains three extra columns: `included`, indicating whether the provider is included based on the `cutoff` argument; `all.events`, indicating if all observations in the provider are 1; `no.events`, indicating if all observations in the provider are 0.

References

He K, Kalbfleisch, J, Li, Y, and et al. (2013) Evaluating hospital readmission rates in dialysis providers; adjusting for hospital effects. *Lifetime Data Analysis*, **19**: 490-512.

Wu, W, Yang, Y, Kang, J, He, K. (2022) Improving large-scale estimation and inference for profiling health care providers. *Statistics in Medicine*, **41(15)**: 2840-2853.

See Also

data_check

Examples

```
data(ExampleDataBinary)
outcome <- ExampleDataBinary$Y
covar <- ExampleDataBinary$Z
ProvID <- ExampleDataBinary$ProvID
data <- data.frame(outcome, ProvID, covar)
covar.char <- colnames(covar)
outcome.char <- colnames(data)[1]
ProvID.char <- colnames(data)[2]
formula <- as.formula(paste("outcome ~", paste(covar.char, collapse = " + "), "+ id(ProvID)"))

# Fit logistic linear effect model using three input formats
fit_fe1 <- logis_fe(Y = outcome, Z = covar, ProvID = ProvID)
fit_fe2 <- logis_fe(data = data, Y.char = outcome.char,
Z.char = covar.char, ProvID.char = ProvID.char)
fit_fe3 <- logis_fe(formula, data)
```

---

logis_firth *Main function for fitting the fixed effect logistic model using firth correction*

Description

Fixed effects (FE) models suffer from separation issues when all outcomes in a cluster are the same, leading to infinite estimates and unreliable inference. Firth's corrected logistic regression (FLR) overcomes this limitation and outperforms both FE and random effects (RE) models in terms of bias and RMSE.

Usage

```
logis_firth(
  formula = NULL,
  data = NULL,
  Y.char = NULL,
```

```
  Z.char = NULL,
  ProvID.char = NULL,
  Y = NULL,
  Z = NULL,
  ProvID = NULL,
  max.iter = 1000,
  tol = 1e-05,
  bound = 10,
  cutoff = 10,
  threads = 1,
  message = TRUE
)
```

Arguments

formula	a two-sided formula object describing the model to be fitted, with the response variable on the left of a ~ operator and covariates on the right, separated by + operators. The fixed effect of the provider identifier is specified using id().
data	a data frame containing the variables named in the formula, or the columns specified by Y.char, Z.char, and ProvID.char.
Y.char	a character string specifying the column name of the response variable in the data.
Z.char	a character vector specifying the column names of the covariates in the data.
ProvID.char	a character string specifying the column name of the provider identifier in the data.
Y	a numeric vector representing the response variable.
Z	a matrix or data frame representing the covariates, which can include both numeric and categorical variables.
ProvID	a numeric vector representing the provider identifier.
max.iter	maximum iteration number if the stopping criterion specified by stop is not satisfied. The default value is 10,000.
tol	tolerance used for stopping the algorithm. See details in stop below. The default value is 1e-5.
bound	a positive number to avoid inflation of provider effects. The default value is 10.
cutoff	An integer specifying the minimum number of observations required for providers. Providers with fewer observations than the cutoff will be labeled as "include = 0" and excluded from model fitting. The default is 10.
threads	a positive integer specifying the number of threads to be used. The default value is 1.
message	a Boolean indicating whether to print the progress of the fitting process. The default is TRUE.

Details

The function accepts three different input formats: a formula and dataset, where the formula is of the form `response ~ covariates + id(provider)`, with `provider` representing the provider identifier; a dataset along with the column names of the response, covariates, and provider identifier; or the binary outcome vector $\boldsymbol{Y}$, the covariate matrix or data frame $\mathbf{Z}$, and the provider identifier vector.

This function utilizes OpenMP for parallel processing. For macOS, to enable multi-threading, users may need to install the OpenMP library (e.g., brew install libomp) or use a supported compiler such as GCC. If OpenMP is not detected during installation, the function will transparently fall back to single-threaded execution.

Value

A list of objects with S3 class `"logis_fe"`:

`coefficient`	a list containing the estimated coefficients: `beta`, the fixed effects for each predictor, and `gamma`, the effect for each provider.
`variance`	a list containing the variance estimates: `beta`, the variance-covariance matrix of the predictor coefficients, and `gamma`, the variance of the provider effects.
`linear_pred`	the linear predictor of each individual.
`fitted`	the predicted probability of each observation having a response of 1.
`observation`	the original response of each individual.
`Loglkd`	the log-likelihood.
`AIC`	Akaike info criterion.
`BIC`	Bayesian info criterion.
`AUC`	area under the ROC curve.
`char_list`	a list of the character vectors representing the column names for the response variable, covariates, and provider identifier. For categorical variables, the names reflect the dummy variables created for each category.
`data_include`	the data used to fit the model, sorted by the provider identifier. For categorical covariates, this includes the dummy variables created for all categories except the reference level. Additionally, it contains three extra columns: `included`, indicating whether the provider is included based on the `cutoff` argument; `all.events`, indicating if all observations in the provider are 1; `no.events`, indicating if all observations in the provider are 0.

References

Firth, D. (1993) Bias reduction of maximum likelihood estimates. *Biometrika*, **80(1)**: 27-38.

See Also

`data_check`

Examples

```
data(ExampleDataBinary)
outcome <- ExampleDataBinary$Y
covar <- ExampleDataBinary$Z
ProvID <- ExampleDataBinary$ProvID
data <- data.frame(outcome, ProvID, covar)
covar.char <- colnames(covar)
outcome.char <- colnames(data)[1]
ProvID.char <- colnames(data)[2]
formula <- as.formula(paste("outcome ~", paste(covar.char, collapse = " + "), "+ id(ProvID)"))

# Fit logistic linear effect model using three input formats
fit_fe1 <- logis_firth(Y = outcome, Z = covar, ProvID = ProvID)
fit_fe2 <- logis_firth(data = data, Y.char = outcome.char,
Z.char = covar.char, ProvID.char = ProvID.char)
fit_fe3 <- logis_firth(formula, data)
```

`logis_re` *Main Function for fitting the random effect logistic model*

Description

Fit a random effect logistic model via `glmer` from the `lme4` package.

Usage

```
logis_re(
  formula = NULL,
  data = NULL,
  Y = NULL,
  Z = NULL,
  ProvID = NULL,
  Y.char = NULL,
  Z.char = NULL,
  ProvID.char = NULL,
  ...
)
```

Arguments

`formula`	a two-sided formula object describing the model to be fitted, with the response variable on the left of a ~ operator and covariates on the right, separated by + operators. The random effect of the provider identifier is specified using `(1 | )`.
`data`	a data frame containing the variables named in the `formula`, or the columns specified by `Y.char`, `Z.char`, and `ProvID.char`.
`Y`	a numeric vector representing the response variable.

Z	a matrix or data frame representing the covariates, which can include both numeric and categorical variables.
ProvID	a numeric vector representing the provider identifier.
Y.char	a character string specifying the column name of the response variable in the data.
Z.char	a character vector specifying the column names of the covariates in the data.
ProvID.char	a character string specifying the column name of the provider identifier in the data.
...	additional arguments passed to glmer for further customization.

Details

This function is used to fit a random effect logistic model of the form:

$$\text{logit}(P(Y_{ij} = 1 \mid \alpha_i, \mathbf{Z}_{ij})) = \mu + \alpha_i + \mathbf{Z}_{ij}^{\top}\boldsymbol{\beta},$$

where Y_{ij} is the binary outcome for individual j in provider i, μ is the overall intercept, α_i is the random effect for provider i, $\mathbf{Z}_{ij}$ are the covariates, and $\boldsymbol{\beta}$ is the vector of coefficients for the covariates.

The model is fitted by overloading the glmer function from the lme4 package. Three different input formats are accepted: a formula and dataset, where the formula is of the form response ~ covariates + (1 | provider), with provider representing the provider identifier; a dataset along with the column names of the response, covariates, and provider identifier; or the outcome vector $\boldsymbol{Y}$, the covariate matrix or data frame $\mathbf{Z}$, and the provider identifier vector.

In addition to these input formats, all arguments from the glmer function can be modified via ..., allowing for customization of model fitting options.

If issues arise during model fitting, consider using the data_check function to perform a data quality check, which can help identify missing values, low variation in covariates, high-pairwise correlation, and multicollinearity. For datasets with missing values, this function automatically removes observations (rows) with any missing values before fitting the model.

Value

A list of objects with S3 class "logis_re":

coefficient	a list containing the estimated coefficients: FE, the fixed effects for each predictor and the intercept, and RE, the random effects for each provider.
variance	a list containing the variance estimates: FE, the variance-covariance matrix of the fixed effect coefficients, and RE, the variance of the random effects.
fitted	the predicted probability of each observation having a response of 1.
observation	the original response of each individual.
linear_pred	the linear predictor of each individual.
data_include	the data used to fit the model, sorted by the provider identifier. For categorical covariates, this includes the dummy variables created for all categories except the reference level.

char_list	a list of the character vectors representing the column names for the response variable, covariates, and provider identifier. For categorical variables, the names reflect the dummy variables created for each category.
Loglkd	the log-likelihood.
AIC	Akaike information criterion.
BIC	Bayesian information criterion.

References

Bates D, Maechler M, Bolker B, Walker S (2015). *Fitting Linear Mixed-Effects Models Using lme4*. Journal of Statistical Software, 67(1), 1-48.

See Also

data_check

Examples

```
data(ExampleDataBinary)
outcome <- ExampleDataBinary$Y
covar <- ExampleDataBinary$Z
ProvID <- ExampleDataBinary$ProvID
data <- data.frame(outcome, ProvID, covar)
covar.char <- colnames(covar)
outcome.char <- colnames(data)[1]
ProvID.char <- colnames(data)[2]
formula <- as.formula(paste("outcome ~", paste(covar.char, collapse = " + "), "+ (1|ProvID)"))

# Fit logistic linear effect model using three input formats
fit_re1 <- logis_re(Y = outcome, Z = covar, ProvID = ProvID)

fit_re2 <- logis_re(data = data, Y.char = outcome.char,
Z.char = covar.char, ProvID.char = ProvID.char)
fit_re3 <- logis_re(formula, data)
```

---

plot.linear_fe *Get funnel plot from a fitted* linear_fe *object for institutional comparisons*

Description

Creates a funnel plot from a linear fixed effect model to compare provider performance.

Usage

```
## S3 method for class 'linear_fe'
plot(
  x,
  null = "median",
  target = 0,
  alpha = 0.05,
  labels = c("lower", "expected", "higher"),
  point_colors = c("#E69F00", "#56B4E9", "#009E73"),
  point_shapes = c(15, 17, 19),
  point_size = 2,
  point_alpha = 0.8,
  line_size = 0.8,
  target_line_type = "longdash",
  ...
)
```

Arguments

x	a model fitted from linear_fe.
null	a character string or a number specifying null hypotheses of fixed provider effects. The default is "median".
target	a numeric value representing the target outcome. The default value is 0.
alpha	a number or a vector of significance levels. The default is 0.05.
labels	a vector of labels for the plot.
point_colors	a vector of colors representing different provider flags. The default is c("#E69F00", "#56B4E9", "#009E73").
point_shapes	a vector of shapes representing different provider flags. The default is c(15, 17, 19).
point_size	size of the points. The default is 2.
point_alpha	transparency level of the points. The default is 0.8.
line_size	size of all lines, including control limits and the target line. The default is 0.8.
target_line_type	line type for the target line. The default is "longdash".
...	additional arguments that can be passed to the function.

Details

This function generates a funnel plot from a linear fixed effect model. Currently, it only supports the indirect standardized difference. The parameter alpha is a vector used to calculate control limits at different significance levels. The first value in the vector is used as the significance level for flagging each provider, utilizing the test.linear_fe function.

Value

A ggplot object representing the funnel plot.

See Also

linear_fe, linear_fe, linear_fe

Examples

```
data(ExampleDataLinear)
outcome <- ExampleDataLinear$Y
covar <- ExampleDataLinear$Z
ProvID <- ExampleDataLinear$ProvID
fit_fe <- linear_fe(Y = outcome, Z = covar, ProvID = ProvID)
plot(fit_fe)
```

---

plot.logis_fe *Get funnel plot from a fitted* logis_fe *object for institutional comparisons*

Description

Creates a funnel plot from a logistic fixed effect model to compare provider performance.

Usage

```
## S3 method for class 'logis_fe'
plot(
  x,
  null = "median",
  test = "score",
  target = 1,
  alpha = 0.05,
  labels = c("lower", "expected", "higher"),
  point_colors = c("#E69F00", "#56B4E9", "#009E73"),
  point_shapes = c(15, 17, 19),
  point_size = 2,
  point_alpha = 0.8,
  line_size = 0.8,
  target_line_type = "longdash",
  ...
)
```

Arguments

x	a model fitted from logis_fe.
null	a character string or a number specifying null hypotheses of fixed provider effects. The default is "median".
test	a character string specifying the type of testing methods to be conducted. The default is "score".

target	a numeric value representing the target outcome. The default value is 1.
alpha	a number or a vector of significance levels. The default is 0.05.
labels	a vector of labels for the plot.
point_colors	a vector of colors representing different provider flags. The default is c("#E69F00", "#56B4E9", "#009E73").
point_shapes	a vector of shapes representing different provider flags. The default is c(15, 17, 19).
point_size	size of the points. The default is 2.
point_alpha	transparency level of the points. The default is 0.8.
line_size	size of all lines, including control limits and the target line. The default is 0.8.
target_line_type	line type for the target line. The default is "longdash".
...	additional arguments that can be passed to the function.

Details

This function generates a funnel plot from a logistic fixed-effect model. Currently, it only supports the indirect standardized ratio. The parameter alpha is a vector used to calculate control limits at different significance levels. The first value in the vector is used as the significance level for flagging each provider, utilizing the test.logis_fe function.

Value

A ggplot object representing the funnel plot.

References

Wu, W., Kuriakose, J. P., Weng, W., Burney, R. E., & He, K. (2023). Test-specific funnel plots for healthcare provider profiling leveraging individual- and summary-level information. *Health Services and Outcomes Research Methodology*, **23(1)**, 45-58.

See Also

logis_fe, SM_output.linear_re, test.logis_fe

Examples

```
data(ExampleDataBinary)
outcome <- ExampleDataBinary$Y
covar <- ExampleDataBinary$Z
ProvID <- ExampleDataBinary$ProvID
fit_fe <- logis_fe(Y = outcome, Z = covar, ProvID = ProvID)
plot(fit_fe)
```

SM_output *Generic function for calculating standardized measures*

Description

SM_output is an S3 generic function designed to calculate standardized measures. It dispatches to the appropriate method based on the class of the input (fit), ensuring the correct method is applied for different types of models.

Usage

```
SM_output(fit, ...)
```

Arguments

fit	the input object, typically a fitted model, for which standardized measures are calculated. The method applied depends on the class of this object.
...	additional arguments that can be passed to specific methods.

Value

the return varies depending on the method implemented for the class of the input object.

SM_output.linear_cre *Calculate direct/indirect standardized differences from a fitted* linear_cre *object*

Description

Provide direct/indirect standardized differences for a correlated random effect linear model.

Usage

```
## S3 method for class 'linear_cre'
SM_output(fit, parm, stdz = "indirect", ...)
```

Arguments

fit	a model fitted from linear_cre.
parm	specifies a subset of providers for which confidence intervals are to be given. By default, all providers are included. The class of parm should match the class of the provider IDs.
stdz	a character string or a vector specifying the standardization method(s). The possible values are: • "indirect" (default) indirect standardization method.

- "direct" direct standardization method.
- c("indirect", "direct") outputs both direct and indirect standardized measures.

... additional arguments that can be passed to the function.

Details

This function computes standardized differences for a random effect linear model using either direct or indirect methods, or both when specified. The function returns both the standardized differences and the observed and expected outcomes used for their calculation.

Value

A list containing the standardized differences based on the method(s) specified in stdz, as well as the observed and expected outcomes used to calculate the standardized measures:

indirect.difference
: indirect standardized differences, if stdz includes "indirect".

direct.difference
: direct standardized differences, if stdz includes "direct".

OE
: a list of data frames containing the observed and expected outcomes used for calculating standardized measures.

Examples

```
data(ExampleDataLinear)
outcome <- ExampleDataLinear$Y
covar <- ExampleDataLinear$Z
ProvID <- ExampleDataLinear$ProvID
data <- data.frame(outcome, ProvID, covar)
outcome.char <- colnames(data)[1]
ProvID.char <- colnames(data)[2]
wb.char <- c("z1", "z2")
other.char <- c("z3", "z4", "z5")
fit_cre <- linear_cre(data = data, Y.char = outcome.char, ProvID.char = ProvID.char,
wb.char = wb.char, other.char = other.char)
SM_output(fit_cre)
```

SM_output.linear_fe *Calculate direct/indirect standardized differences from a fitted* linear_fe *object*

Description

Provide direct/indirect standardized differences for a fixed effect linear model.

Usage

```
## S3 method for class 'linear_fe'
SM_output(fit, parm, stdz = "indirect", null = "median", ...)
```

Arguments

`fit`	a model fitted from `linear_fe`.
`parm`	specifies a subset of providers for which confidence intervals are to be given. By default, all providers are included. The class of `parm` should match the class of the provider IDs.
`stdz`	a character string or a vector specifying the standardization method(s). The possible values are: • `"indirect"` (default) indirect standardization method. • `"direct"` direct standardization method. • `c("indirect", "direct")` outputs both direct and indirect standardized measures.
`null`	a character string or a number defining the population norm. The default value is `"median"`. The possible values are: • `"median"` the median of the provider effect estimates ($\hat{\gamma}_i$). • `"mean"` the weighted average of the provider effect estimates ($\hat{\gamma}_i$), where the weights correspond to the sample size of each provider. • numeric a user-defined numeric value representing the population norm.
`...`	additional arguments that can be passed to the function.

Details

This function computes standardized differences for a fixed effect linear model using either direct or indirect methods, or both when specified. For each method, the population norm is determined by the `null` argument. The population norm can be the median of the estimates, their weighted mean (with weights corresponding to provider sizes), or a user-defined numeric value.

Value

A list containing the standardized differences based on the method(s) specified in `stdz`, as well as the observed and expected outcomes used to calculate the standardized measures:

`indirect.difference`	indirect standardized differences, if `stdz` includes `"indirect"`.
`direct.difference`	direct standardized differences, if `stdz` includes `"direct"`.
`OE`	a list of data frames containing the observed and expected outcomes used for calculating standardized measures.

Examples

```
data(ExampleDataLinear)
outcome <- ExampleDataLinear$Y
covar <- ExampleDataLinear$Z
ProvID <- ExampleDataLinear$ProvID
fit_linear <- linear_fe(Y = outcome, Z = covar, ProvID = ProvID)
SM_output(fit_linear)
SM_output(fit_linear, stdz = "direct", null = "mean")
```

SM_output.linear_re *Calculate direct/indirect standardized differences from a fitted* linear_re *object*

Description

Provide direct/indirect standardized differences for a random effect linear model.

Usage

```
## S3 method for class 'linear_re'
SM_output(fit, parm, stdz = "indirect", ...)
```

Arguments

fit	a model fitted from linear_re.
parm	specifies a subset of providers for which confidence intervals are to be given. By default, all providers are included. The class of parm should match the class of the provider IDs.
stdz	a character string or a vector specifying the standardization method(s). The possible values are: • "indirect" (default) indirect standardization method. • "direct" direct standardization method. • c("indirect", "direct") outputs both direct and indirect standardized measures.
...	additional arguments that can be passed to the function.

Details

This function computes standardized differences for a random effect linear model using either direct or indirect methods, or both when specified. The function returns both the standardized differences and the observed and expected outcomes used for their calculation.

Value

A list containing the standardized differences based on the method(s) specified in `stdz`, as well as the observed and expected outcomes used to calculate the standardized measures:

`indirect.difference`
: indirect standardized differences, if `stdz` includes `"indirect"`.

`direct.difference`
: direct standardized differences, if `stdz` includes `"direct"`.

`OE`
: a list of data frames containing the observed and expected outcomes used for calculating standardized measures.

Examples

```
data(ExampleDataLinear)
outcome <- ExampleDataLinear$Y
covar <- ExampleDataLinear$Z
ProvID <- ExampleDataLinear$ProvID
fit_linear <- linear_re(Y = outcome, Z = covar, ProvID = ProvID)
SM_output(fit_linear)
```

SM_output.logis_cre *Calculate direct/indirect standardized ratios/rates from a fitted* `logis_cre` *object*

Description

Provide direct/indirect standardized ratios/rates for a correlated random effect logistic model.

Usage

```
## S3 method for class 'logis_cre'
SM_output(
  fit,
  parm,
  stdz = "indirect",
  measure = c("rate", "ratio"),
  threads = 2,
  ...
)
```

Arguments

`fit`
: a model fitted from `logis_cre`.

`parm`
: specifies a subset of providers for which confidence intervals are to be given. By default, all providers are included. The class of `parm` should match the class of the provider IDs.

stdz	a character string or a vector specifying the standardization method(s). The possible values are: • "indirect" (default) indirect standardization method. • "direct" direct standardization method. • c("indirect", "direct") outputs both direct and indirect standardized measures.
measure	a character string or a vector indicating whether the output measure is "ratio" or "rate" • "rate" output the standardized rate. The "rate" has been restricted to 0% - 100%. • "ratio" output the standardized ratio. • c("ratio", "rate") (default) output both the ratio and rate.
threads	an integer specifying the number of threads to use. The default value is 2.
...	additional arguments that can be passed to the function.

Value

A list contains standardized measures, as well as the observed and expected outcomes used for calculation, depending on the user's choice of standardization method (stdz) and measure type (measure).

indirect.ratio	standardization ratio using indirect method if stdz includes "indirect" and measure includes "ratio".
direct.ratio	standardization ratio using direct method if stdz includes "direct" and measure includes "ratio".
indirect.rate	standardization rate using indirect method if stdz includes "indirect" and measure includes "rate".
direct.rate	standardization rate using direct method if stdz includes "direct" and measure includes "rate".
OE	a list of data frames containing the observed and expected outcomes used for calculating standardized measures.

Examples

```
data(ExampleDataBinary)
outcome <- ExampleDataBinary$Y
covar <- ExampleDataBinary$Z
ProvID <- ExampleDataBinary$ProvID
data <- data.frame(outcome, ProvID, covar)
outcome.char <- colnames(data)[1]
ProvID.char <- colnames(data)[2]
wb.char <- c("z1", "z2")
other.char <- c("z3", "z4", "z5")
fit_cre <- logis_cre(data = data, Y.char = outcome.char, ProvID.char = ProvID.char,
wb.char = wb.char, other.char = other.char)

SR <- SM_output(fit_cre, stdz = "direct", measure = "rate")
```

```
SR$direct.rate
```

`SM_output.logis_fe` *Calculate direct/indirect standardized ratios/rates from a fitted* `logis_fe` *object*

Description

Provide direct/indirect standardized ratios/rates for a fixed effect logistic model.

Usage

```
## S3 method for class 'logis_fe'
SM_output(
  fit,
  parm,
  stdz = "indirect",
  measure = c("rate", "ratio"),
  null = "median",
  threads = 2,
  ...
)
```

Arguments

`fit`	a model fitted from `logis_fe`.
`parm`	specifies a subset of providers for which confidence intervals are to be given. By default, all providers are included. The class of `parm` should match the class of the provider IDs.
`stdz`	a character string or a vector specifying the standardization method(s). The possible values are: • `"indirect"` (default) indirect standardization method. • `"direct"` direct standardization method. • `c("indirect", "direct")` outputs both direct and indirect standardized measures.
`measure`	a character string or a vector indicating whether the output measure is "ratio" or "rate" • `"rate"` output the standardized rate. The "rate" has been restricted to 0% - 100%. • `"ratio"` output the standardized ratio. • `c("ratio", "rate")` (default) output both the ratio and rate.
`null`	if `"stdz = indirect"`, a character string or a number defining the population norm. The default is "median".
`threads`	an integer specifying the number of threads to use. The default value is 2.
`...`	additional arguments that can be passed to the function.

Value

A list contains standardized measures, as well as the observed and expected outcomes used for calculation, depending on the user's choice of standardization method (`stdz`) and measure type (`measure`).

`indirect.ratio`	standardization ratio using indirect method if `stdz` includes `"indirect"` and `measure` includes `"ratio"`.
`direct.ratio`	standardization ratio using direct method if `stdz` includes `"direct"` and `measure` includes `"ratio"`.
`indirect.rate`	standardization rate using indirect method if `stdz` includes `"indirect"` and `measure` includes `"rate"`.
`direct.rate`	standardization rate using direct method if `stdz` includes `"direct"` and `measure` includes `"rate"`.
`OE`	a list of data frames containing the observed and expected outcomes used for calculating standardized measures.

References

He, K. (2019). Indirect and direct standardization for evaluating transplant centers. *Journal of Hospital Administration*, **8(1)**, 9-14.

Examples

```
data(ExampleDataBinary)
outcome = ExampleDataBinary$Y
covar = ExampleDataBinary$Z
ProvID = ExampleDataBinary$ProvID
fit_fe <- logis_fe(Y = outcome, Z = covar, ProvID = ProvID, message = FALSE)
SR <- SM_output(fit_fe, stdz = "direct", measure = "rate")
SR$direct.rate
```

`SM_output.logis_re` *Calculate direct/indirect standardized ratios/rates from a fitted* `logis_re` *object*

Description

Provide direct/indirect standardized ratios/rates for a random effect logistic model.

Usage

```
## S3 method for class 'logis_re'
SM_output(
  fit,
  parm,
  stdz = "indirect",
  measure = c("rate", "ratio"),
  threads = 2,
  ...
)
```

Arguments

fit	a model fitted from logis_re.
parm	specifies a subset of providers for which confidence intervals are to be given. By default, all providers are included. The class of parm should match the class of the provider IDs.
stdz	a character string or a vector specifying the standardization method(s). The possible values are: • "indirect" (default) indirect standardization method. • "direct" direct standardization method. • c("indirect", "direct") outputs both direct and indirect standardized measures.
measure	a character string or a vector indicating whether the output measure is "ratio" or "rate" • "rate" output the standardized rate. The "rate" has been restricted to 0% - 100%. • "ratio" output the standardized ratio. • c("ratio", "rate") (default) output both the ratio and rate.
threads	an integer specifying the number of threads to use. The default value is 2.
...	additional arguments that can be passed to the function.

Value

A list contains standardized measures, as well as the observed and expected outcomes used for calculation, depending on the user's choice of standardization method (stdz) and measure type (measure).

indirect.ratio	standardization ratio using indirect method if stdz includes "indirect" and measure includes "ratio".
direct.ratio	standardization ratio using direct method if stdz includes "direct" and measure includes "ratio".
indirect.rate	standardization rate using indirect method if stdz includes "indirect" and measure includes "rate".

direct.rate	standardization rate using direct method if stdz includes "direct" and measure includes "rate".
OE	a list of data frames containing the observed and expected outcomes used for calculating standardized measures.

Examples

```
data(ExampleDataBinary)
outcome = ExampleDataBinary$Y
covar = ExampleDataBinary$Z
ProvID = ExampleDataBinary$ProvID
fit_re <- logis_re(Y = outcome, Z = covar, ProvID = ProvID)
SR <- SM_output(fit_re, stdz = "direct", measure = "rate")
SR$direct.rate
```

---

summary.linear_cre *Result Summaries of Covariate Estimates from a fitted* linear_fe*,* linear_re *or* linear_cre *object*

Description

Provide the summary statistics for the covariate estimates for a fixed/random/correlated random effect linear model.

Usage

```
## S3 method for class 'linear_cre'
summary(object, parm, level = 0.95, null = 0, ...)

## S3 method for class 'linear_fe'
summary(object, parm, level = 0.95, null = 0, ...)

## S3 method for class 'linear_re'
summary(object, parm, level = 0.95, null = 0, ...)
```

Arguments

object	a model fitted from linear_fe or linear_re or linear_cre.
parm	specifies a subset of covariates for which the result summaries should be output. By default, all covariates are included.
level	the confidence level during the hypothesis test, meaning a significance level of $1 -$ level. The default value is 0.95.
null	a number defining the null hypothesis for the covariate estimates. The default value is 0.
...	additional arguments that can be passed to the function.

Value

A data frame containing summary statistics for covariate estimates, with the following columns:

Estimate	the estimates of covariate coefficients.
Std.Error	the standard error of the estimate.
Stat	the test statistic.
p value	the p-value for the hypothesis test.
CI.upper	the lower bound of the confidence interval.
CI.lower	the upper bound of the confidence interval.

Examples

```
data(ExampleDataLinear)
outcome <- ExampleDataLinear$Y
covar <- ExampleDataLinear$Z
ProvID <- ExampleDataLinear$ProvID
data <- data.frame(outcome, ProvID, covar)
outcome.char <- colnames(data)[1]
ProvID.char <- colnames(data)[2]
wb.char <- c("z1", "z2")
other.char <- c("z3", "z4", "z5")
fit_cre <- linear_cre(data = data, Y.char = outcome.char, ProvID.char = ProvID.char,
wb.char = wb.char, other.char = other.char)
summary(fit_cre)

data(ExampleDataLinear)
outcome <- ExampleDataLinear$Y
covar <- ExampleDataLinear$Z
ProvID <- ExampleDataLinear$ProvID
fit_fe <- linear_fe(Y = outcome, Z = covar, ProvID = ProvID)
summary(fit_fe)

data(ExampleDataLinear)
outcome <- ExampleDataLinear$Y
covar <- ExampleDataLinear$Z
ProvID <- ExampleDataLinear$ProvID
fit_re <- linear_fe(Y = outcome, Z = covar, ProvID = ProvID)
summary(fit_re)
```

---

summary.logis_cre *Result Summaries of Covariate Estimates from a fitted* logis_re *or* logis_cre *object*

Description

Provide the summary statistics for the covariate estimates for a random/correlated random effect logistic model.

Usage

```
## S3 method for class 'logis_cre'
summary(object, parm, level = 0.95, null = 0, ...)

## S3 method for class 'logis_re'
summary(object, parm, level = 0.95, null = 0, ...)
```

Arguments

object	a model fitted from logis_re or logis_cre.
parm	specifies a subset of covariates for which the result summaries should be output. By default, all covariates are included.
level	the confidence level during the hypothesis test, meaning a significance level of $1 -$ level. The default value is 0.95.
null	a number defining the null hypothesis for the covariate estimates. The default value is 0.
...	additional arguments that can be passed to the function.

Value

A data frame containing summary statistics for covariate estimates, with the following columns:

Estimate	the estimates of covariate coefficients.
Std.Error	the standard error of the estimate.
Stat	the test statistic.
p value	the p-value for the hypothesis test.
CI.upper	the lower bound of the confidence interval.
CI.lower	the upper bound of the confidence interval.

Examples

```
data(ExampleDataBinary)
outcome <- ExampleDataBinary$Y
covar <- ExampleDataBinary$Z
ProvID <- ExampleDataBinary$ProvID
data <- data.frame(outcome, ProvID, covar)
outcome.char <- colnames(data)[1]
ProvID.char <- colnames(data)[2]
wb.char <- c("z1", "z2")
other.char <- c("z3", "z4", "z5")
fit_cre <- logis_cre(data = data, Y.char = outcome.char, ProvID.char = ProvID.char,
wb.char = wb.char, other.char = other.char)
summary(fit_cre)


data(ExampleDataBinary)
outcome <- ExampleDataBinary$Y
covar <- ExampleDataBinary$Z
```

```
ProvID <- ExampleDataBinary$ProvID
fit_re <- logis_re(Y = outcome, Z = covar, ProvID = ProvID)
summary(fit_re)
```

`summary.logis_fe` *Result Summaries of Covariate Estimates from a fitted* `logis_fe` *object*

Description

Provide the summary statistics for the covariate estimates for a fixed effect logistic model.

Usage

```
## S3 method for class 'logis_fe'
summary(object, parm, level = 0.95, test = "wald", null = 0, ...)
```

Arguments

`object`	a model fitted from `logis_fe`.
`parm`	Specifies a subset of covariates for which the result summaries should be output. By default, all covariates are included.
`level`	the confidence level during the hypothesis test, meaning a significance level of $1 -$ level. The default value is 0.95.
`test`	a character string specifying the type of testing method. The default is "wald". • `"wald"`: wald test. • `"lr"`: likelihood ratio test. • `"score"`: score test.
`null`	a number defining the null hypothesis for the covariate estimates. The default value is `0`.
`...`	additional arguments that can be passed to the function.

Value

A data frame containing summary statistics for covariate estimates, with the following columns:

`Estimate`	the estimates of covariate coefficients.
`Std.Error`	the standard error of the estimate, included only when `test = "wald"`.
`Stat`	the test statistic.
`p value`	the p-value for the hypothesis test.
`CI.upper`	the lower bound of the confidence interval, included only when `test = "wald"`.
`CI.lower`	the upper bound of the confidence interval, included only when `test = "wald"`.

Examples

```
data(ExampleDataBinary)
outcome = ExampleDataBinary$Y
covar = ExampleDataBinary$Z
ProvID = ExampleDataBinary$ProvID

fit_fe <- logis_fe(Y = outcome, Z = covar, ProvID = ProvID, message = FALSE)
summary.wald <- summary(fit_fe, level = 0.95, test = "wald")
summary.wald
```

---

test *Generic function for hypothesis testing of provider effects*

Description

`test` is an S3 generic function used to conduct hypothesis tests on provider effect coefficients and detect outlying provider. The function dispatches to the appropriate method based on the class of the input model (`fit`).

Usage

```
test(fit, ...)
```

Arguments

`fit`	the input object, typically a fitted model, for which provider effects are tested. The method applied depends on the class of this object.
`...`	additional arguments that can be passed to specific methods.

Value

the return depends on the method implemented for the class of the input object, typically including statistical outputs for provider effect coefficients and identification of outlier providers.

---

test.linear_cre *Conduct hypothesis testing for provider effects from a fitted* `linear_cre` *object*

Description

Conduct hypothesis tests on provider effects and identify outlying providers for a correlated random effect linear model.

Usage

```
## S3 method for class 'linear_cre'
test(fit, parm, level = 0.95, null = 0, alternative = "two.sided", ...)
```

Arguments

fit	a model fitted from linear_cre.
parm	specifies a subset of providers for which confidence intervals are to be given. By default, all providers are included. The class of parm should match the class of the provider IDs.
level	the confidence level during the hypothesis test, meaning a significance level of $1 -$ level. The default value is 0.95.
null	a number defining the null hypothesis for the provider effects. The default value is 0.
alternative	a character string specifying the alternative hypothesis, must be one of "two.sided" (default), "greater", or "less".
...	additional arguments that can be passed to the function.

Details

The function identifies outlying providers based on hypothesis test results. For two-sided tests, 1 indicates performance significantly higher than expected, -1 indicates lower, For one-sided tests, 1 (right-tailed) or -1 (left-tailed) flags are used. Providers whose performance falls within the central range are flagged as 0. Outlying providers are determined by the test statistic falling beyond the threshold based on the significance level $1 -$ level.

Value

A data frame containing the results of the hypothesis test, with the following columns:

flag	a flagging indicator where 1 means statistically higher than expected and -1 means statistically lower than expected.
p-value	the p-value of the hypothesis test.
stat	the test statistic.
Std.Error	the standard error of the provider effect estimate.

Examples

```
data(ExampleDataLinear)
outcome <- ExampleDataLinear$Y
covar <- ExampleDataLinear$Z
ProvID <- ExampleDataLinear$ProvID
data <- data.frame(outcome, ProvID, covar)
outcome.char <- colnames(data)[1]
ProvID.char <- colnames(data)[2]
wb.char <- c("z1", "z2")
other.char <- c("z3", "z4", "z5")
```

```
fit_cre <- linear_cre(data = data, Y.char = outcome.char, ProvID.char = ProvID.char,
wb.char = wb.char, other.char = other.char)
test(fit_cre)
```

`test.linear_fe` *Conduct hypothesis testing for provider effects from a fitted* `linear_fe` *object*

Description

Conduct hypothesis tests on provider effects and identify outlying providers for a fixed effect linear model.

Usage

```
## S3 method for class 'linear_fe'
test(fit, parm, level = 0.95, null = "median", alternative = "two.sided", ...)
```

Arguments

`fit`	a model fitted from `linear_fe`.
`parm`	specifies a subset of providers for which confidence intervals are to be given. By default, all providers are included. The class of `parm` should match the class of the provider IDs.
`level`	the confidence level during the hypothesis test, meaning a significance level of $1 -$ level. The default value is 0.95.
`null`	a character string or a number defining the null hypothesis for the provider effects. The default value is `"median"`. The possible values are: • `"median"`: The median of the provider effect estimates ($\hat{\gamma}_i$). • `"mean"`: The weighted average of the provider effect estimates ($\hat{\gamma}_i$), where the weights correspond to the sample size of each provider. • numeric: A user-defined numeric value representing the null hypothesis.
`alternative`	a character string specifying the alternative hypothesis, must be one of `"two.sided"` (default), `"greater"`, or `"less"`.
`...`	additional arguments that can be passed to the function.

Details

The function identifies outlying providers based on hypothesis test results. For two-sided tests, `1` indicates performance significantly higher than expected, `-1` indicates lower, For one-sided tests, `1` (right-tailed) or `-1` (left-tailed) flags are used. Providers whose performance falls within the central range are flagged as `0`. Outlying providers are determined by the test statistic falling beyond the threshold based on the significance level $1 -$ level.

Value

A data frame containing the results of the hypothesis test, with the following columns:

`flag`	a flagging indicator where `1` means statistically higher than expected and `-1` means statistically lower than expected.
`p-value`	the p-value of the hypothesis test.
`stat`	the test statistic.
`Std.Error`	the standard error of the provider effect estimate.

Examples

```
data(ExampleDataLinear)
outcome <- ExampleDataLinear$Y
covar <- ExampleDataLinear$Z
ProvID <- ExampleDataLinear$ProvID
fit_linear <- linear_fe(Y = outcome, Z = covar, ProvID = ProvID)
test(fit_linear)
```

---

test.linear_re *Conduct hypothesis testing for provider effects from a fitted* `linear_re` *object*

Description

Conduct hypothesis tests on provider effects and identify outlying providers for a random effect linear model.

Usage

```
## S3 method for class 'linear_re'
test(fit, parm, level = 0.95, null = 0, alternative = "two.sided", ...)
```

Arguments

`fit`	a model fitted from `linear_re`.
`parm`	specifies a subset of providers for which confidence intervals are to be given. By default, all providers are included. The class of `parm` should match the class of the provider IDs.
`level`	the confidence level during the hypothesis test, meaning a significance level of $1 -$ level. The default value is 0.95.
`null`	a number defining the null hypothesis for the provider effects. The default value is 0.
`alternative`	a character string specifying the alternative hypothesis, must be one of `"two.sided"` (default), `"greater"`, or `"less"`.
`...`	additional arguments that can be passed to the function.

Details

The function identifies outlying providers based on hypothesis test results. For two-sided tests, `1` indicates performance significantly higher than expected, `-1` indicates lower, For one-sided tests, `1` (right-tailed) or `-1` (left-tailed) flags are used. Providers whose performance falls within the central range are flagged as `0`. Outlying providers are determined by the test statistic falling beyond the threshold based on the significance level $1 -$ level.

Value

A data frame containing the results of the hypothesis test, with the following columns:

`flag`	a flagging indicator where `1` means statistically higher than expected and `-1` means statistically lower than expected.
`p-value`	the p-value of the hypothesis test.
`stat`	the test statistic.
`Std.Error`	the standard error of the provider effect estimate.

Examples

```
data(ExampleDataLinear)
outcome <- ExampleDataLinear$Y
ProvID <- ExampleDataLinear$ProvID
covar <- ExampleDataLinear$Z
fit_re <- linear_re(Y = outcome, Z = covar, ProvID = ProvID)
test(fit_re)
```

---

test.logis_cre *Conduct hypothesis testing for provider effects from a fitted* `logis_cre` *object*

Description

Conduct hypothesis tests on provider effects and identify outlying providers for a correlated random effect logistic model.

Usage

```
## S3 method for class 'logis_cre'
test(fit, parm, level = 0.95, null = 0, alternative = "two.sided", ...)
```

Arguments

fit	a model fitted from logis_cre.
parm	specifies a subset of providers for which confidence intervals are to be given. By default, all providers are included. The class of parm should match the class of the provider IDs.
level	the confidence level during the hypothesis test, meaning a significance level of $1 -$ level. The default value is 0.95.
null	a number defining the null hypothesis for the provider effects. The default value is 0.
alternative	a character string specifying the alternative hypothesis, must be one of "two.sided" (default), "greater", or "less".
...	additional arguments that can be passed to the function.

Details

The function identifies outlying providers based on hypothesis test results. For two-sided tests, 1 indicates performance significantly higher than expected, -1 indicates lower, For one-sided tests, 1 (right-tailed) or -1 (left-tailed) flags are used. Providers whose performance falls within the central range are flagged as 0. Outlying providers are determined by the test statistic falling beyond the threshold based on the significance level $1 -$ level.

Value

A data frame containing the results of the hypothesis test, with the following columns:

flag	a flagging indicator where 1 means statistically higher than expected and -1 means statistically lower than expected.
p-value	the p-value of the hypothesis test.
stat	the test statistic.
Std.Error	the standard error of the provider effect estimate.

Examples

```
data(ExampleDataBinary)
outcome <- ExampleDataBinary$Y
covar <- ExampleDataBinary$Z
ProvID <- ExampleDataBinary$ProvID
data <- data.frame(outcome, ProvID, covar)
outcome.char <- colnames(data)[1]
ProvID.char <- colnames(data)[2]
wb.char <- c("z1", "z2")
other.char <- c("z3", "z4", "z5")
fit_cre <- logis_cre(data = data, Y.char = outcome.char, ProvID.char = ProvID.char,
wb.char = wb.char, other.char = other.char)
test(fit_cre)
```

`test.logis_fe` *Conduct hypothesis testing for provider effects from a fitted* `logis_fe` *object*

Description

Conduct hypothesis tests on provider effects and identify outlying providers for a fixed effect logistic model.

Usage

```
## S3 method for class 'logis_fe'
test(
  fit,
  parm,
  level = 0.95,
  test = "exact.poisbinom",
  score_modified = TRUE,
  null = "median",
  n = 10000,
  threads = 1,
  alternative = "two.sided",
  ...
)
```

Arguments

`fit`	a model fitted from `logis_fe`.
`parm`	specifies a subset of providers for which confidence intervals are to be given. By default, all providers are included. The class of `parm` should match the class of the provider IDs.
`level`	the confidence level during the hypothesis test, meaning a significance level of $1 -$ level. The default value is 0.95.
`test`	a character string specifying the type of testing method to be conducted. The default is "exact.poisbinom". • `"exact.poisbinom"`: exact test based on Poisson-binomial distribution of $O_i \| Z_i$. • `"exact.bootstrap"`: exact test based on bootstrap procedure. • `"wald"`: wald test. • `"score"`: score test.
`score_modified`	a logical indicating whether to use the modified score test ignoring the randomness of covariate coefficient for score teat (`"test = score"`). The default value is TRUE.
`null`	a character string or a number specifying null hypotheses of fixed provider effects. The default is `"median"`.

n	resample size for bootstrapping when ("test = exact.bootstrap"). The default value is 10,000.
threads	an integer specifying the number of threads to use. The default value is 1.
alternative	a character string specifying the alternative hypothesis, must be one of "two.sided" (default), "greater", or "less".
...	additional arguments that can be passed to the function.

Details

By default, the function uses the "exact.poisbinom" method. The wald test is invalid for extreme providers (i.e. when provider effect goes to infinity). For the score test, consider that when the number of tested providers is large, refitting the models to get the restricted MLEs will take a long time. Therefore, we use unrestricted MLEs to replace the restricted MLEs during the testing procedure by default. However, the user can specify score_modified = FALSE to perform a standard score test.

Value

A data frame containing the results of the hypothesis test, with the following columns:

flag	a flagging indicator where 1 means statistically higher than expected and -1 means statistically lower than expected.
p-value	the p-value of the hypothesis test.
stat	the test statistic.
Std.Error	The standard error of the provider effect estimate, included only when test = "wald".

References

Wu, W, Yang, Y, Kang, J, He, K. (2022) Improving large-scale estimation and inference for profiling health care providers. *Statistics in Medicine*, **41(15)**: 2840-2853.

Examples

```
data(ExampleDataBinary)
outcome = ExampleDataBinary$Y
covar = ExampleDataBinary$Z
ProvID = ExampleDataBinary$ProvID
fit_fe <- logis_fe(Y = outcome, Z = covar, ProvID = ProvID, message = FALSE)
test(fit_fe, test = "score")
```

`test.logis_re` *Conduct hypothesis testing for provider effects from a fitted* `logis_re` *object*

Description

Conduct hypothesis tests on provider effects and identify outlying providers for a random effect logistic model.

Usage

```
## S3 method for class 'logis_re'
test(fit, parm, level = 0.95, null = 0, alternative = "two.sided", ...)
```

Arguments

`fit`	a model fitted from `logis_re`.
`parm`	specifies a subset of providers for which confidence intervals are to be given. By default, all providers are included. The class of `parm` should match the class of the provider IDs.
`level`	the confidence level during the hypothesis test, meaning a significance level of $1 -$ level. The default value is 0.95.
`null`	a number defining the null hypothesis for the provider effects. The default value is 0.
`alternative`	a character string specifying the alternative hypothesis, must be one of `"two.sided"` (default), `"greater"`, or `"less"`.
`...`	additional arguments that can be passed to the function.

Details

The function identifies outlying providers based on hypothesis test results. For two-sided tests, `1` indicates performance significantly higher than expected, `-1` indicates lower, For one-sided tests, `1` (right-tailed) or `-1` (left-tailed) flags are used. Providers whose performance falls within the central range are flagged as `0`. Outlying providers are determined by the test statistic falling beyond the threshold based on the significance level $1 -$ level.

Value

A data frame containing the results of the hypothesis test, with the following columns:

`flag`	a flagging indicator where `1` means statistically higher than expected and `-1` means statistically lower than expected.
`p-value`	the p-value of the hypothesis test.
`stat`	the test statistic.
`Std.Error`	the standard error of the provider effect estimate.

Examples

```
data(ExampleDataBinary)
outcome <- ExampleDataBinary$Y
ProvID <- ExampleDataBinary$ProvID
covar <- ExampleDataBinary$Z
fit_re <- logis_re(Y = outcome, Z = covar, ProvID = ProvID)
test(fit_re)
```

Index

∗ **datasets**
ecls_data, 16
ExampleDataBinary, 17
ExampleDataLinear, 18

bar_plot, 3

caterpillar_plot, 4
confint.linear_cre, 6
confint.linear_fe, *6*, 8
confint.linear_re, *6*, 9
confint.logis_cre, 10
confint.logis_fe, *6*, 12
confint.logis_re, 14

data_check, 15, *20*, *23*, *25*, *27*, *31*, *33*, *36*

ecls_data, 16
ExampleDataBinary, 17
ExampleDataLinear, 18

glmer, *26*, *34, 35*

linear_cre, 19
linear_fe, 21, *38*
linear_re, 23
lmer, *19*, *23, 24*
logis_cre, 26
logis_fe, 28, *39*
logis_firth, 31
logis_re, 34

plot.linear_fe, 36
plot.logis_fe, 38

SM_output, 40
SM_output.linear_cre, 40
SM_output.linear_fe, *8*, 41
SM_output.linear_re, *7*, *10, 11*, *14*, *39*, 43
SM_output.logis_cre, 44
SM_output.logis_fe, *13*, 46
SM_output.logis_re, 47
summary.linear_cre, 49
summary.linear_fe *(*summary.linear_cre*)*, 49
summary.linear_re *(*summary.linear_cre*)*, 49
summary.logis_cre, 50
summary.logis_fe, 52
summary.logis_re *(*summary.logis_cre*)*, 50

test, 53
test.linear_cre, 53
test.linear_fe, *4*, *37*, 55
test.linear_re, *4*, 56
test.logis_cre, 57
test.logis_fe, *4*, *39*, 59
test.logis_re, 61