Package ‘CLRtools’

March 16, 2026

Title Diagnostic Tools for Logistic and Conditional Logistic Regression

Version 0.1.1

Description Provides tools for fitting, assessing, and comparing logistic and conditional logistic regression models. Includes residual diagnostics and goodness of fit measures for model development and evaluation in matched case control studies.

License GPL-3

Encoding UTF-8

RoxygenNote 7.3.3

Imports bayesplot, caret, dplyr, ggplot2, ggpubr, lmtest, loo, patchwork, RColorBrewer, rlang, rstan, survival, tidyr

Suggests dagitty, ggdag, knitr, rmarkdown, rstanarm

VignetteBuilder knitr

Depends R (>= 3.5)

LazyData true

URL `https://github.com/brendacontla/CLRtools`

BugReports `https://github.com/brendacontla/CLRtools/issues`

NeedsCompilation no

Author Brenda Contla Hernández [aut, cre],
Matthieu Vignes [ctb] (Provided statistical guidance, conceptual feedback, and supervision),
Chris Compton [ctb] (Provided statistical guidance, conceptual feedback, and supervision)

Maintainer Brenda Contla Hernández `<B.Hernandez@massey.ac.nz>`

Repository CRAN

Date/Publication 2026-03-16 08:00:28 UTC

Contents

`check_coef_change` *Assess Coefficient Change After Variable Removal*

Description

Computes the percentage change in logistic regression coefficients ($\Delta\hat{\beta}\%$) as each additional variable is introduced to the model one at a time. Supports both standard logistic regression (via `glm`) and conditional logistic regression (via `clogit`) when a stratification variable is provided.

Usage

```
check_coef_change(data, yval, xpre, xcheck, strata = NULL)
```

Arguments

`data`	A data frame containing the outcome, predictors, and optional stratification variable.
`yval`	A string naming the binary outcome variable.
`xpre`	A character vector of variable names that are already selected for the model.
`xcheck`	A character vector of variable names to be added one-by-one for comparison.
`strata`	Optional; a string specifying the name of the stratification variable for conditional logistic regression.

Details

This function fits a logistic regression model using variables in `xpre`, and then adds each variable in `xcheck` one at a time to assess how the coefficients of the model change with the delta beta hat percentages. Useful for evaluating confounding or additional variable contribution. When `strata` is `NULL`, the function uses standard logistic regression (`glm` with binomial family). When `strata` is specified, conditional logistic regression is used instead via `survival::clogit`.

Value

A data frame showing how the coefficients change when each variable in `xcheck` is added to the model containing `xpre`.

References

Hosmer, D. W., Lemeshow, S., & Sturdivant, R. X. (2013). *Applied Logistic Regression* (3rd ed.). John Wiley & Sons, Inc. The formulas for calculating residuals and diagnostics are adapted from this source.

See Also

`delta.coefficient`

Examples

```
# Example from Hosmer et al., 2013
# Applied Logistic Regression (3rd ed.), Chapter 4

# Variables selected to evaluate
preliminar <- c('age', 'height', 'priorfrac', 'momfrac', 'armassist')

# Variable to evaluate for potential confounding
excluded <- c('raterisk')

# Assess coefficient change after adding 'raterisk'
check_coef_change(data = glow500, yval = 'fracture', xpre = preliminar, xcheck = excluded)
```

`check_coef_significant`

Check Significance of Excluded Variables

Description

Obtain summary statistics, including Wald test z-values and p-values, for coefficients of variables added one at a time to an existing logistic regression model. Supports both standard and conditional logistic regression.

Usage

```
check_coef_significant(data, yval, xpre, xcheck, strata = NULL)
```

Arguments

`data`	A data frame containing the outcome, predictors, and optionally a stratification variable.
`yval`	A string naming the binary outcome variable.
`xpre`	A character vector of variables included in the preliminary model.
`xcheck`	A character vector of variables to be tested for significance when added individually to the preliminary model.
`strata`	(Optional) A string naming the stratification variable. If provided, conditional logistic regression is used via `clogit()`.

Details

For each variable in `xcheck`, this function fits a model that includes the variable alongside `xpre`. It extracts and returns the coefficient summary for each added variable to assess statistical significance. When `strata` is provided, a conditional logistic regression model is used instead of standard logistic regression.

Value

A matrix containing the coefficient estimates, standard errors, z-values, and p-values for each variable in `xcheck` when added to the preliminary model.

Examples

```
# Example from Hosmer et al., 2013
# Applied Logistic Regression (3rd ed.), Chapter 4

# Variables selected for the full model
preliminar <- c('age', 'height', 'priorfrac', 'momfrac', 'armassist', 'raterisk')

# Variables to test
excluded <- c('weight', 'bmi', 'premeno', 'smoke')
```

```
# Assess whether any excluded variables become significant when added to the preliminary model
check_coef_significant(data = glow500, yval = 'fracture', xpre = preliminar, xcheck = excluded)
```

check_interactions *Check Pairwise Interactions in Logistic Regression*

Description

Evaluates all pairwise interactions between provided predictor variables in a logistic regression model. It compares models that include individual interaction terms to a main-effects model using likelihood ratio tests. Supports both standard logistic regression and conditional logistic regression (CLR) when stratification is provided.

Usage

```
check_interactions(data, variables, yval, rounding = NULL, strata = NULL)
```

Arguments

`data`	A data frame containing the outcome variable and predictor variables.
`variables`	A character vector of predictor variables to test for pairwise interactions.
`yval`	A string naming the binary outcome variable.
`rounding`	Optional integer specifying the number of decimal places to round the output. Defaults to `NULL` (no rounding).
`strata`	A string naming the stratification variable for conditional logistic regression (CLR). Defaults to `NULL`, meaning standard logistic regression is used.

Details

This function fits a logistic regression model using the variables in `variables`, then iteratively fits additional models that include each pairwise interaction. The models are compared using likelihood ratio tests via `lmtest::lrtest`. If `strata` is provided, conditional logistic regression (CLR) is used instead of standard logistic regression.

Value

A data frame showing the log-likelihood, likelihood ratio test statistic (G), and p-value for each pairwise interaction tested. The first row corresponds to the main-effects model.

Examples

```
# Example from Hosmer et al., 2013
# Applied Logistic Regression (3rd ed.), Chapter 4, Table 4.14

# Evaluate potential interaction terms among predictors.
# Variables included in the final main effects model to evaluate potential interactions
var.names <- c('age', 'height', 'priorfrac', 'momfrac', 'armassist', 'raterisk_cat')

# Recode 'raterisk' into a binary categorical variable 'raterisk_cat'
glow500<-dplyr::mutate(
  glow500,
  raterisk_cat = dplyr::case_when(
    raterisk %in% c('Less', 'Same') ~ 'C1',
    raterisk == 'Greater' ~ 'C2'))

# Run the interaction-checking procedure
check_interactions(data = glow500, variables = var.names, yval = 'fracture', rounding = 4)
```

`coeff.OR` *Compute Odds Ratios for Logistic and Conditional Logistic Regression Models*

Description

Calculates odds ratios and confidence intervals for one or more coefficients in a fitted logistic regression model (from `glm`) or conditional logistic regression model (from `clogit`). Also supports scaling coefficients to meaningful units (e.g., per 5 years, 5 kg, or 5 cm) for clearer interpretation.

Usage

```
coeff.OR(model, variable = NULL, c = 1, confidence.level = 0.95)
```

Arguments

`model`	A fitted model object from `glm()` (with `family = binomial`) or `survival::clogit()`.
`variable`	Optional character vector specifying the variable(s) for which to compute odds ratios. If `NULL` (default), checks all coefficients
`c`	A numeric value specifying the increment(s) for the variables. Either a single (unnamed) numeric value applied to all selected variables, or a named numeric vector specifying a separate increment for each. The names of `c` must match the coefficient names in the model. Defaults to 1.
`confidence.level`	Confidence level for the confidence interval. Defaults to 0.95.

Details

Supports both logistic regression models fit with `glm()` and conditional logistic regression models fit with `clogit()` from the `survival` package. Confidence intervals are calculated on the log-odds scale and transformed back using `exp()`.

Value

A named list containing:

`odds.ratio`	Exponentiated coefficients (odds ratios).
`lower.limit`	Lower bounds of the confidence intervals.
`upper.limit`	Upper bounds of the confidence intervals.

Examples

```
# Example from Hosmer et al., 2013
# Applied Logistic Regression (3rd ed.), Chapter 5, Table 5.18

# Recode 'raterisk' into a binary categorical variable 'raterisk_cat'
glow500<-dplyr::mutate(
  glow500,
  raterisk_cat = dplyr::case_when(
    raterisk %in% c('Less', 'Same') ~ 'C1',
    raterisk == 'Greater' ~ 'C2'))

 model.int <- glm(
   fracture ~ age + height + priorfrac + momfrac +
     armassist + raterisk_cat + age*priorfrac + momfrac*armassist,
  family = binomial, data = glow500)

 # Specify variables and interpretation units for OR computation
 var.or<-c('raterisk_catC2','height')
 units.var<-c('raterisk_catC2'=1,'height'=5)

 # Calculate and interpret adjusted odds ratios
 coeff.OR(model.int,variable = var.or, c = units.var)
```

`compare_bayesm` *Posterior Predictive Check for Multiple Bayesian Models*

Description

This function performs a posterior predictive check for one or more Bayesian models by computing the mean and standard deviations of simulated predictions, comparing them to the observed outcome. It produces grouped histograms for each model.

Usage

```
compare_bayesm(
  models,
  ypredict = NULL,
  data,
  outcome,
  intercept = NULL,
  var.param = NULL
)
```

Arguments

`models`	A **named list** of fitted `rstan` models (objects of class `stanfit`).
`ypredict`	Optional. A named list of posterior predictive matrices. Each matrix should have rows as posterior draws and columns as data points. If not provided, predictions are computed internally. Must have the same names as `models`.
`data`	A data frame containing the predictor variables and the outcome used for model prediction.
`outcome`	A character string naming the outcome variable in `data`.
`intercept`	(Optional) A named list with the intercept parameter names for each model. Each entry should be a character string or `NULL` if no intercept is used. Must have the same names as `models`.
`var.param`	A **named list** mapping each model to a named character vector where each name is a variable in the data and the value is the name of the corresponding parameter/coefficient in the model. Must have the same names as `models`. Required if `ypredict` is not provided.

Value

Invisibly returns a list with two ggplot objects:

ppc_pe A ggplot object showing the distribution of simulated mean of the outcome across models.

ppc_sd A ggplot object showing the distribution of simulated standard deviations of the outcome across models.

---

compare_bayesm_by_predictor

Compare Bayesian Models by Predictor Using Posterior Predictive Simulations

---

Description

This function compares posterior predictive distributions from several Bayesian models across levels of a selected predictor variable. For numeric predictors, the variable is binned; for categorical predictors, the original factor levels are used directly. The function visualizes the distribution of simulated means and standard deviations per predictor level alongside the observed values.

Usage

```
compare_bayesm_by_predictor(
  data,
  models,
  parameters = NULL,
  var.plot,
  intercept = NULL,
  ypredict = NULL,
  outcome,
  mbreaks
)
```

Arguments

`data`	A data frame containing the original dataset.
`models`	A named list of fitted `stanfit` model objects. Each name will be used as the model label.
`parameters`	Optional. A **named list** mapping each model to a named character vector where each name is a variable in the data and the value is the name of the corresponding parameter/coefficient in the model. Must have the same names as `models`. Required if `ypredict` is not provided.
`var.plot`	A single character string. Name of the predictor variable in `data` used for binning and plotting. Must be in all the models.
`intercept`	Optional. A named list with the intercept parameter names for each model. Each entry should be a character string or `NULL` if no intercept is used. Must have the same names as `models`.
`ypredict`	Optional. A named list of posterior predictive matrices. Each matrix should have rows as posterior draws and columns as data points. If not provided, predictions are computed internally. Must have the same names as `models`.
`outcome`	A character string. The name of the outcome variable in `data`.
`mbreaks`	Number of bins if `var.plot` is numeric; ignored if it's a factor.

Details

This function provides a visual diagnostic for comparing posterior predictive summaries across multiple Bayesian models. The predictor variable can be either continuous or categorical. For continuous variables, the range is divided into bins using `cut()` and `mbreaks`; for categorical variables, no binning is applied. Posterior predictive distributions are either precomputed via `ypredict` or generated internally using `parameters` and (optionally) `intercept`.

Value

A list containing:

models_summary A data frame summarizing the posterior predictive means and standard deviations per draw, model, and predictor level (or bin).

p_mean A `ggplot` showing the distribution of simulated outcome means for each bin and model, overlaid with the observed means and sample size per bin.

p_sd A `ggplot` showing the distribution of simulated outcome standard deviations for each bin and model, overlaid with the observed standard deviations and sample size per bin.

The function also prints the plots side by side using `ggarrange()`.

`compare_models_loo` *Compare Bayesian Models Using PSIS-LOO*

Description

This function compares multiple Bayesian models using PSIS-LOO (Pareto-smoothed importance sampling leave-one-out cross-validation) from the `loo` package. It returns a comparison table and a plot of the estimated ELPD (expected log predictive density) with standard errors.

Usage

```
compare_models_loo(..., k = 0.7, name_log = "log_lik")
```

Arguments

`...`	Two or more `stanfit` model objects to compare. Each model must include pointwise log-likelihood values (usually named `log_lik`) stored in the generated quantities or transformed parameters block.
`k`	A numeric value specifying the Pareto-k diagnostic threshold. Default is 0.7.
`name_log`	A character string specifying the name of the log-likelihood parameter in the model. Default is `"log_lik"`.

Details

This function performs PSIS-LOO diagnostics on each model, creates a visual summary, and ranks them using `loo_compare`. Ensure that each model includes pointwise log-likelihood values named consistently (e.g., `"log_lik"`).

Value

A list with the following elements:

`p_loo` A ggplot object showing `elpd_loo` values and standard errors for each model.

`comparison` A `loo_compare` table comparing the relative fit of the models.

`models_loo` A named list of individual `loo` objects for each model.

confidence.interval *Compute Wald-Based Confidence Intervals for Logit and Predicted Probability*

Description

Calculates Wald-based confidence intervals for the predicted logit values and estimated probabilities for new data points, based on a fitted logistic regression model.

Usage

```
confidence.interval(model, data, confidence.level = 0.95)
```

Arguments

`model`	A fitted logistic regression model from `glm()` with `family` = `binomial`.
`data`	A data frame containing the new observations for which predictions and confidence intervals are needed.
`confidence.level`	Confidence level for the interval estimates. Defaults to 0.95.

Value

A list with two elements:

`logit` A data frame with predicted logit values and their lower and upper confidence limits.

`estimated.prob` A data frame with predicted probabilities and their lower and upper confidence limits.

Examples

```
# Example from Hosmer et al., 2013
# Applied Logistic Regression (3rd ed.), Chapter 2

# Fit logistic regression model with selected predictors, Table 2.3
mod2.3 <- glm(
  fracture ~ age + priorfrac + raterisk,
  family = binomial,
  data = glow500
)

# Create a new data point for prediction
new.x <- data.frame(
  age = 65,
  priorfrac = "Yes",
  raterisk = "Same"
)
```

```
# Compute the 95% confidence interval for the predicted probability
confidence.interval(mod2.3, data = new.x, confidence.level = 0.95)
```

`cov.patterns` *Extract Unique Covariate Patterns from a Logistic Regression Model*

Description

Returns a summary of unique covariate patterns from a fitted logistic regression model, including the number of observations per pattern and the estimated probability.

Usage

```
cov.patterns(model)
```

Arguments

`model`	A fitted logistic regression model object, typically from `glm()` with `family = binomial`.

Details

This function summarizes the unique covariate patterns in the data, capturing the combinations of predictor values across observations. It calculates the frequency of each pattern, the number of events (cases), and the estimated probability for each combination of covariates.

Value

A data frame where each row corresponds to a unique covariate pattern. The output includes:

`y_j` The number of observed events (cases) for each pattern.

`m` The number of observations sharing that pattern.

`est.prob` The estimated probability from the fitted model for that pattern.

`Covariate columns` The covariates used in the model, showing the pattern structure.

Examples

```
# Example from Hosmer et al., 2013
# Applied Logistic Regression (3rd ed.), Chapter 5

# Recode 'raterisk' into a binary variable 'raterisk_cat'
glow500 <- dplyr::mutate(
  glow500,
  raterisk_cat = dplyr::case_when(
    raterisk %in% c("Less", "Same") ~ "C1",
    raterisk == "Greater" ~ "C2"
  )
)

# Fit a multiple logistic regression model with interactions
model.int <- glm(
  fracture ~ age + height + priorfrac + momfrac + armassist +
    raterisk_cat + age * priorfrac + momfrac * armassist,
  family = binomial,
  data = glow500
)

# Examine covariate patterns from the fitted model
X.cv <- cov.patterns(model.int)
head(X.cv, n=10)
```

`cutpoints` *Table with Sensitivity and Specificity at Different Cutpoints*

Description

This function computes the sensitivity and specificity at various cutpoints for a given logistic regression model. It generates a table summarizing the performance metrics (sensitivity, specificity) at different probability cutoffs and optionally plots these metrics and the distribution of probabilities for each class. This is useful for selecting an optimal threshold for classification.

Usage

```
cutpoints(model, cmin = 0, cmax = 1, byval = 0.05, plot = TRUE)
```

Arguments

`model`	A fitted logistic regression model (either `glm` or `clogit`).
`cmin`	The minimum cutoff value for the predicted probabilities. Defaults to 0.
`cmax`	The maximum cutoff value for the predicted probabilities. Defaults to 1.
`byval`	The increment for cutpoints. Defaults to 0.05.
`plot`	Logical value indicating whether to generate plots. Defaults to TRUE.

Details

The function calculates sensitivity and specificity for a range of cutpoints from `cmin` to `cmax` with a step size of `byval`. It then plots the relationship between sensitivity and specificity, as well as histograms of estimated probabilities. The cutpoint with the smallest difference between sensitivity and specificity is also marked on the histogram plots. This can aid in finding an optimal classification threshold.

Value

A data frame containing cutpoints, sensitivity, specificity, and specificity complement for each cutoff. If `plot = TRUE`, a ggplot2-based visualization is also printed, showing sensitivity and specificity curves and the distribution of predicted probabilities by outcome class, with the optimal cutoff (where sensitivity and specificity are closest) indicated on the histogram.

Examples

```
# Example from Hosmer et al., 2013
# Applied Logistic Regression (3rd ed.), Chapter 5, Table 5.8

# Recode 'raterisk' into a binary variable 'raterisk_cat'
glow500 <- dplyr::mutate(
  glow500,
  raterisk_cat = dplyr::case_when(
    raterisk %in% c("Less", "Same") ~ "C1",
    raterisk == "Greater" ~ "C2"
  )
)

# Fit a multiple logistic regression model with interactions
model.int <- glm(
  fracture ~ age + height + priorfrac + momfrac + armassist +
    raterisk_cat + age * priorfrac + momfrac * armassist,
  family = binomial,
  data = glow500
)

# Compute sensitivity and specificity at multiple cutpoints
cutpoints(model.int, cmin = 0.05, cmax = 0.75, byval = 0.05, plot = FALSE)
```

`delta.coefficient` *Delta-beta hat percentage: Change in Coefficients when Adding a Variable*

Description

This function checks the percentage change in the coefficient of a variable when it is added to a model. The comparison is made between two models: model1 (with the new variable included) and model0 (without the new variable).

Usage

```
delta.coefficient(model1, model0, variable = NULL)
```

Arguments

`model1`	A fitted model (e.g., `glm`, `lm`) with the new variable included.
`model0`	A fitted model (e.g., `glm`, `lm`) without the new variable.
`variable`	A vector of variable names to check. If `NULL` (default), checks all coefficients.

Details

This function compares the coefficients of variables between two logistic regression models or conditional logistic regressions: one that includes a variable of interest (`model1`) and one that excludes it (`model0`). It calculates how much the coefficients of other variables change, expressed as a percentage. The change is calculated as ((beta0 - beta1) / beta1) * 100, where beta0 is the coefficient from model0 and beta1 is the coefficient from model1.

Value

A vector of percentage changes in the coefficients between model1 and model0.

Examples

```
# Example from Hosmer et al., 2013
# Applied Logistic Regression (3rd ed.), Chapter 3, Table 3.10

mod3.1 <- glm(fracture ~ priorfrac, family = binomial, data = glow500)
mod3.2 <- glm(fracture ~ priorfrac + height, family = binomial, data = glow500)

# delta-beta-coefficient
delta.coefficient(model1 = mod3.2, model0 = mod3.1, variable = 'priorfrac')
```

`diagnosticplots_class` *Diagnostic Plots for Model Discrimination*

Description

Generates four diagnostic plots to evaluate the discriminatory ability of a logistic regression model. These plots follow the style of those presented by Hosmer et al. (2013) and are helpful for visually assessing model performance.

Usage

```
diagnosticplots_class(model)
```

Arguments

`model`	A fitted logistic regression model object of class `glm`, with a binary outcome.

Details

This function creates the following diagnostic plots:

1. A jitter plot showing the distribution of the estimated probabilities against the observed outcome.
2. An ROC curve displaying sensitivity versus 1-specificity.
3. A histogram of estimated probabilities for observations with outcome = 0.
4. A histogram of estimated probabilities for observations with outcome = 1.

Value

This function displays a 2x2 grid of diagnostic plots.

References

Hosmer, D. W., Lemeshow, S., & Sturdivant, R. X. (2013). *Applied Logistic Regression* (3rd ed.). John Wiley & Sons, Inc.

See Also

`cutpoints`

Examples

```
# Example from Hosmer et al., 2013
# Applied Logistic Regression (3rd ed.), Chapter 5, Section 5.2.4

# Recode 'raterisk' into a binary variable 'raterisk_cat'
glow500 <- dplyr::mutate(
  glow500,
  raterisk_cat = dplyr::case_when(
    raterisk %in% c("Less", "Same") ~ "C1",
    raterisk == "Greater" ~ "C2"
  )
)

# Fit a multiple logistic regression model with interactions
model.int <- glm(
  fracture ~ age + height + priorfrac + momfrac + armassist +
    raterisk_cat + age * priorfrac + momfrac * armassist,
  family = binomial,
  data = glow500
)

# Generate classification diagnostic plots
diagnosticplots_class(model.int)
```

`diagnostic_bayes` *Generate MCMC Diagnostic Plots for a Bayesian Model*

Description

Produces a set of standard diagnostic plots to evaluate convergence and sampling efficiency for a Bayesian model fitted using `rstan`.

Usage

```
diagnostic_bayes(model, var.param)
```

Arguments

`model`	A fitted Bayesian model object created with `rstan::stan()`.
`var.param`	A character vector specifying the names of the model parameters (e.g., slopes or coefficients) to include in the trace and rank plots.

Details

This function uses the `bayesplot` package to visualize diagnostics commonly used to assess convergence and sampling performance in MCMC estimation.

Value

A list of four ggplot2 objects:

plot_rhat A histogram of Rhat values (`mcmc_rhat_hist`).

plot_neff A histogram of effective sample sizes (`mcmc_neff_hist`).

plot_trace Trace plots of the MCMC chains for selected parameters (`mcmc_trace`).

plot_trank Rank plots of the selected parameters across chains (`mcmc_rank_overlay`).

`discordant.pairs` *Count Discordant Pairs in Matched Case-Control Data*

Description

This function identifies and counts discordant/non-informative pairs for categorical variables within matched case-control data. It supports flexible matching designs, including 1:1 and 1:m, by evaluating each stratum individually. Only strata with both case and control observations are included in the evaluation.

Usage

```
discordant.pairs(data, outcome, strata, variables = NULL)
```

Arguments

`data`	A data frame containing the matched dataset.
`outcome`	A character string giving the name of the binary outcome variable.
`strata`	A character string giving the name of the variable that identifies matched strata or sets.
`variables`	Optional. A character vector of variable names to evaluate for discordant pairs. If `NULL`, all categorical (factor or character) variables in `data` are used.

Value

A named numeric vector:

- Each element corresponds to the number of discordant strata for one variable.
- The final element, `total.valid.pairs`, gives the total number of strata that were eligible for evaluation.

Examples

```
# Example from Hosmer et al., 2013
# Applied Logistic Regression (3rd ed.), Chapter 7, Table 7.1

discordant.pairs(glow11m, outcome = 'fracture', strata = 'pair')
```

`DRtest` *Deviance Residuals Test (HL Test)*

Description

The function performs the Hosmer-Lemeshow (HL) goodness-of-fit test, which is used to evaluate how well a logistic regression model fits the data by comparing observed and expected frequencies across different groups based on predicted probabilities. This function calculates the test using either the model's predictions or user-supplied predictions. It divides the data into g groups, based on the predicted values, and calculates the chi-squared statistic to assess the fit.

Usage

```
DRtest(model = NULL, yvar = NULL, yhatvar = NULL, g = 10)
```

Arguments

`model`	A fitted logistic regression model from `glm()` with `family = binomial`. If model is supplied, the yvar and yhatvar arguments will not be used.
`yvar`	A vector of observed response values. Required if `model` is `NULL`.
`yhatvar`	A vector of predicted probabilities. Required if `model` is `NULL`.
`g`	The number of groups (quantiles) to divide the data into for the Hosmer-Lemeshow test. Default is 10.

Details

The Hosmer-Lemeshow test compares the observed and expected frequencies of events across different quantiles of predicted probabilities. The test statistic follows a chi-squared distribution.

Value

A list containing the following components:

`results`	A data frame with observed and expected values, and total counts for each group. The data frame includes columns for the observed and expected counts for both the outcome (1 and 0) and total counts.
`chisq`	The chi-squared statistic used to assess the fit of the model.
`df`	The degrees of freedom for the chi-squared test.
`p.value`	The p-value of the chi-squared test, which assesses the overall goodness of fit.
`groups`	The number of groups (quantiles) used in the test.

Examples

```
# Example from Hosmer et al., 2013
# Applied Logistic Regression (3rd ed.), Chapter 5, Table 5.2

# Recode 'raterisk' into a binary variable 'raterisk_cat'
glow500 <- dplyr::mutate(
  glow500,
  raterisk_cat = dplyr::case_when(
    raterisk %in% c("Less", "Same") ~ "C1",
    raterisk == "Greater" ~ "C2"
  )
)

# Fit a multiple logistic regression model with interactions
model.int <- glm(
  fracture ~ age + height + priorfrac + momfrac + armassist +
    raterisk_cat + age * priorfrac + momfrac * armassist,
  family = binomial,
  data = glow500
)

# Perform Hosmer-Lemeshow test with default 10 groups
DRtest(model.int)
```

`glow11m` *GLOW11M dataset*

Description

This dataset was originally shared by Hosmer et al. (2013).

Usage

```
glow11m
```

Format

A data frame with 238 rows and 16 variables:

pair Matched pair identifier (each case-control pair shares the same value)

sub_id Identification Code

site_id Study Site (1–6)

phy_id Physician ID code (128 unique codes)

priorfrac History of prior fracture (1 = Yes, 0 = No)

age Years

weight Kilograms

height Centimeters

bmi Body mass index (kg/m^2)

premeno Menopause before age 45 (1 = Yes, 0 = No)

momfrac Mother had hip fracture (1 = Yes, 0 = No)

armassist Arms are needed to stand from a chair (1 = Yes, 0 = No)

smoke Former or current smoker (1 = Yes, 0 = No)

raterisk Self-reported risk of fracture (1 = Less than others of the same age, 2 = Same as others of the same age, 3 = Greater than others of the same age)

fracscore Composite risk score

fracture Any fracture in first year (1 = Yes, 0 = No)

Source

Dataset shared by Hosmer et al. (2013) with permission. Also appears in the **aplore3** package.

`glow500` *GLOW500 dataset*

Description

This dataset was originally shared by Hosmer et al. (2013).

Usage

```
glow500
```

Format

A data frame with 500 rows and 15 variables:

sub_id Identification Code

site_id Study Site (1–6)

phy_id Physician ID code (128 unique codes)

priorfrac History of prior fracture (1 = Yes, 0 = No)

age Years

weight Kilograms

height Centimeters

bmi Body mass index (kg/m^2)

premeno Menopause before age 45 (1 = Yes, 0 = No)

momfrac Mother had hip fracture (1 = Yes, 0 = No)

armassist Arms are needed to stand from a chair (1 = Yes, 0 = No)

smoke Former or current smoker (1 = Yes, 0 = No)

raterisk Self-reported risk of fracture (1 = Less than others of the same age, 2 = Same as others of the same age, 3 = Greater than others of the same age)

fracscore Composite risk score

fracture Any fracture in first year (1 = Yes, 0 = No)

Source

Dataset shared by Hosmer et al. (2013) with permission. Also appears in the **aplore3** package.

`logit_prob_plot` *Plot Predicted Probabilities from a Logistic Model*

Description

This function visualizes the predicted probabilities from a Bayesian logistic regression model fitted using `rstan`. It computes the posterior predicted probabilities over a grid defined by two continuous predictor variables, and creates a plot showing how these probabilities vary across their values. Color is used to represent the estimated probability, and the original data points are overlaid for reference.

Usage

```
logit_prob_plot(
  data,
  ypredict = NULL,
  model = NULL,
  parameters = NULL,
  intercept = NULL,
  outcome,
  predictors.plot
)
```

Arguments

`data`	A data frame containing the original data used to fit the model.
`ypredict`	Optional. A matrix of posterior predictive simulations of the outcome variable (e.g., generated externally). If `NULL`, predictions will be simulated internally assuming a single-level logistic regression. The matrix should have dimensions `S` x `N`, where `S` is the number of posterior draws (rows) and `N` is the number of observations (columns).
`model`	A fitted Stan model object of class `'stanfit'` (from `rstan`). Required if `ypredict` is not provided.
`parameters`	A named vector where the names are the predictor variable names (as in the data), and the values are the corresponding parameter names in the Stan model. Required if `ypredict` is not provided.
`intercept`	Optional. A character string indicating the name of the intercept parameter in the Stan model (if present).
`outcome`	A character string with the name of the binary outcome variable in the data.
`predictors.plot`	A character vector of length 2 specifying which two predictor variables to use for the x and y axes of the plot.

Value

A `ggplot2` object showing the mean predicted probabilities across the grid of the two specified predictors, with the observed outcome overlaid as colored points.

`osius_rojek` *Osius and Rojek Goodness-of-Fit Test for Logistic Regression*

Description

Applies the Osius and Rojek normal approximation to assess the overall goodness-of-fit of a logistic regression model. This includes:

- A normal approximation to the distribution of the Pearson chi-square statistic.
- A normal approximation to the distribution of the sum-of-squares statistic.

Usage

```
osius_rojek(model)
```

Arguments

`model`	A logistic regression model fitted with `glm()`.

Details

This function implements the Osius and Rojek test as described in Hosmer et al. (2013) for assessing the goodness-of-fit of logistic regression models.

Value

A list containing the following measures:

`z_chisq`	The standardized Z-statistic based on the Pearson chi-square approximation.
`p_value`	The two-sided p-value associated with `z_chisq`.
`z_s`	The standardized Z-statistic based on the sum-of-squares approximation.
`p_value_S`	The two-sided p-value associated with `z_s`.

References

Hosmer, D. W., Lemeshow, S., & Sturdivant, R. X. (2013). *Applied Logistic Regression* (3rd ed.). John Wiley & Sons, Inc.

See Also

`cov.patterns`

Examples

```
# Example from Hosmer et al., 2013
# Applied Logistic Regression (3rd ed.), Chapter 5, Section 5.2.2

# Recode 'raterisk' into a binary variable 'raterisk_cat'
glow500 <- dplyr::mutate(
  glow500,
  raterisk_cat = dplyr::case_when(
    raterisk %in% c("Less", "Same") ~ "C1",
    raterisk == "Greater" ~ "C2"
  )
)

# Fit a multiple logistic regression model with interactions
model.int <- glm(
  fracture ~ age + height + priorfrac + momfrac + armassist +
    raterisk_cat + age * priorfrac + momfrac * armassist,
  family = binomial,
  data = glow500
)

# Apply the Osius and Rojek test for goodness-of-fit
osius_rojek(model.int)
```

rcv_measures *Model Fit Evaluation: R^2-like Measures for Logistic Regression Models with J < n*

Description

This function computes several R^2-like measures for logistic regression models, including the squared Pearson correlation coefficient, pseudo R-squared, adjusted R-squared, shrinkage-adjusted R-squared, and log-likelihood-based R-squared. These metrics, adapted from Hosmer et al. (2013), are specifically designed for datasets where the number of covariates (J) is smaller than the number of data points (n), providing a comprehensive assessment of model fit in such contexts.

Usage

```
rcv_measures(model)
```

Arguments

`model`	A fitted logistic regression model of class `glm`.

Value

A list containing the following measures:

`squared.Pearson.cc`	The squared Pearson correlation coefficient.
`R_ss`	The linear regression-like measure or pseudo R-squared value.
`ajusted_R`	The adjusted R-squared value, adjusted for the number of covariates and sample size.
`adjust_shrink_R`	The shrinkage-adjusted R-squared value, which accounts for potential shrinkage in model estimates.
`R_LS`	The log-likelihood-based R-squared value.

References

Hosmer, D. W., Lemeshow, S., & Sturdivant, R. X. (2013). *Applied Logistic Regression* (3rd ed.). John Wiley & Sons, Inc.

See Also

`cov.patterns`, `residuals_logistic`

Examples

```
# Example from Hosmer et al., 2013
# Applied Logistic Regression (3rd ed.), Chapter 5, Section 5.2.5

# Recode 'raterisk' into a binary variable 'raterisk_cat'
glow500 <- dplyr::mutate(
  glow500,
  raterisk_cat = dplyr::case_when(
    raterisk %in% c("Less", "Same") ~ "C1",
    raterisk == "Greater" ~ "C2"
  )
)

# Fit a multiple logistic regression model with interactions
model.int <- glm(
  fracture ~ age + height + priorfrac + momfrac + armassist +
    raterisk_cat + age * priorfrac + momfrac * armassist,
  family = binomial,
  data = glow500
)

# Compute R-squared-like model fit statistics assuming J < n
rcv_measures(model.int)
```

`residuals_clog` *Model Diagnostic for Conditional Logistic Regression*

Description

This function computes various diagnostic measures for a conditional logistic regression model, including Pearson residuals, leverage, standardized Pearson residuals, delta chi, and delta beta values. Additionally, it generates plots to visualize these diagnostics, helping to assess the model's fit and identify potential influential data points. This function assumes that the `strata` and `y` vectors are provided in the same order as they were used when fitting the model with the `clogit` function.

Usage

```
residuals_clog(model, strata, y, plot = TRUE)
```

Arguments

`model`	A fitted conditional logistic regression model of class `clogit`.
`strata`	A vector of the strata (matching variables) that defines the pairs in the matched case-control design.
`y`	A vector of the binary outcome variable (case-control indicator, where 1 represents the case and 0 represents the control).
`plot`	A logical value indicating whether diagnostic plots should be generated. Defaults to TRUE.

Details

The diagnostic measures are calculated based on the formulas provided by Hosmer et al. (2013). The following measures are computed:

- Pearson residuals
- Standardized Pearson residuals
- Leverage
- Delta statistics: Delta beta (change in Cook's distance) and Delta chi (change in Pearson chi-squared)

Additionally, the function generates the following plots:

- Leverage vs. Estimated Probability
- Change in Pearson chi-squared vs. Estimated Probability
- Cook's Distance vs. Estimated Probability
- Change in Pearson chi-squared and Change in Cook's Distance vs Pair ID

Value

A list containing:

`data.results`	A data frame with the calculated residuals, leverage, and other diagnostic measures for each observation.
`leverage`	A ggplot object displaying leverage vs. estimated probability.
`change.Pearsonchi`	A ggplot object displaying the change in Pearson chi-squared vs. estimated probability.
`cooks`	A ggplot object displaying Cook's distance vs. estimated probability.
`m.Pearsonchi`	A ggplot object displaying the total pairwise change in Pearson chi-squared.
`mcooks`	A ggplot object displaying the total pairwise change in Cook's distance.

References

Hosmer, D. W., Lemeshow, S., & Sturdivant, R. X. (2013). *Applied Logistic Regression* (3rd ed.). John Wiley & Sons, Inc.

Examples

```
# Example from Hosmer et al., 2013
# Applied Logistic Regression (3rd ed.), Chapter 7

# Convert 'fracture' to binary (0 = No, 1 = Yes)
glow11m$fracture <- ifelse(glow11m$fracture == "Yes", 1, 0)

# Load required package
library(survival)

# Fit a conditional logistic regression model
```

```
mod7.3 <- clogit(
  fracture ~ weight + bmi + priorfrac + momfrac + armassist + strata(pair),
  data = glow11m)

# Run diagnostics for the conditional logistic model
residuals_clog(mod7.3, strata = glow11m$pair, y = glow11m$fracture)
```

---

residuals_logistic *Model Diagnostic for Logistic Regression Models*

Description

This function computes various residuals and diagnostic measures for logistic regression models, including Pearson residuals, standardized Pearson residuals, deviance residuals, leverage, and delta statistics (change in Pearson chi-squared, change in deviance, and change in Cook's distance). The function also generates diagnostic plots for evaluating the model fit and identifying influential data points.

Usage

```
residuals_logistic(model)
```

Arguments

`model`	A fitted logistic regression model of class `glm`.

Details

The residuals and diagnostic measures are computed using standard formulas for logistic regression (adapted from Hosmer et al., 2013, *Applied Logistic Regression*, 3rd ed.). The following measures are computed:

- Pearson residuals
- Standardized Pearson residuals
- Deviance residuals
- Leverage
- Delta statistics: Delta beta (change in Cook's distance), Delta chi (change in Pearson chi-squared) and Delta deviance

Additionally, the function generates the following plots:

- Leverage vs. Estimated Probability
- Change in Pearson chi-squared vs. Estimated Probability
- Change in Deviance vs. Estimated Probability
- Cook's Distance vs. Estimated Probability
- Change in Pearson chi-squared vs. Estimated Probability
- Change in Pearson chi-squared vs. Estimated Probability with size determinate by Cook's distance.

Value

A list containing:

`x.cv`	A data frame with the computed residuals and diagnostic measures for each observation, and their respective covariates.
`leverage`	A ggplot object displaying leverage vs. estimated probability.
`change.Pearsonchi`	A ggplot object displaying the change in Pearson chi-squared vs. estimated probability.
`change.deviance`	A ggplot object displaying the change in deviance vs. estimated probability.
`cooks`	A ggplot object displaying Cook's distance vs. estimated probability.
`change.Pb`	A ggplot object displaying the change in Pearson chi-squared vs. Estimated Probability with size determinate by Cook's distance.

References

Hosmer, D. W., Lemeshow, S., & Sturdivant, R. X. (2013). *Applied Logistic Regression* (3rd ed.). John Wiley & Sons, Inc. The formulas for calculating residuals and diagnostics are adapted from this source.

See Also

`cov.patterns`

Examples

```
# Example from Hosmer et al., 2013
# Applied Logistic Regression (3rd ed.), Chapter 5, Section 5.3

# Recode 'raterisk' into a binary variable 'raterisk_cat'
glow500 <- dplyr::mutate(
  glow500,
  raterisk_cat = dplyr::case_when(
    raterisk %in% c("Less", "Same") ~ "C1",
    raterisk == "Greater" ~ "C2"
  )
)

# Fit a multiple logistic regression model with interactions
model.int <- glm(
  fracture ~ age + height + priorfrac + momfrac + armassist +
    raterisk_cat + age * priorfrac + momfrac * armassist,
  family = binomial,
  data = glow500
)

# Run diagnostics for the logistic model
residuals_logistic(model.int)
```

`r_measures` *Model Fit Evaluation: R^2-like Measures for Logistic Regression Models with J=n*

Description

This function computes several R^2-like measures for logistic regression models, including the squared Pearson correlation coefficient, pseudo R-squared, adjusted R-squared, shrinkage-adjusted R-squared, and log-likelihood-based R-squared. These metrics, adapted from Hosmer et al. (2013), are specifically designed for datasets where the number of covariate patterns (J) is equal to the number of data points (n), offering a comprehensive assessment of model fit in such contexts.

Usage

```
r_measures(model)
```

Arguments

`model`	A fitted logistic regression model of class `glm`.

Value

A list containing the following measures:

`squared.Pearson.cc`	The squared Pearson correlation coefficient.
`R_ss`	The linear regression-like measure or pseudo R-squared value.
`ajusted_R`	The adjusted R-squared value, adjusted for the number of covariates and sample size.
`adjust_shrink_R`	The shrinkage-adjusted R-squared value, which accounts for potential shrinkage in model estimates.
`R_L`	The log-likelihood-based R-squared value.

References

Hosmer, D. W., Lemeshow, S., & Sturdivant, R. X. (2013). *Applied Logistic Regression* (3rd ed.). John Wiley & Sons, Inc.

Examples

```
# Example from Hosmer et al., 2013
# Applied Logistic Regression (3rd ed.), Chapter 5, Section 5.2.5

# Recode 'raterisk' into a binary variable 'raterisk_cat'
glow500 <- dplyr::mutate(
  glow500,
  raterisk_cat = dplyr::case_when(
    raterisk %in% c("Less", "Same") ~ "C1",
    raterisk == "Greater" ~ "C2"
  )
)

# Fit a multiple logistic regression model with interactions
model.int <- glm(
  fracture ~ age + height + priorfrac + momfrac + armassist +
    raterisk_cat + age * priorfrac + momfrac * armassist,
  family = binomial,
  data = glow500
)

# Compute R-squared-like model fit statistics assuming J = n
r_measures(model.int)
```

`stukels_test` *Stukel's Test for Logistic Regression Model Fit*

Description

This function performs Stukel's test to assess the goodness-of-fit of a logistic regression model, as adapted from Hosmer et al. (2013) by using the likelihood ratio test .

Usage

```
stukels_test(model)
```

Arguments

`model`	A fitted logistic regression model of class `glm`.

Value

A list with the following components:

`G`	Likelihood ratio test statistic for added z1 and z2.
`p_value`	P-value for the likelihood ratio test.
`model_summary`	Summary of the logistic regression model including z1 and z2.

References

Hosmer, D. W., Lemeshow, S., & Sturdivant, R. X. (2013). *Applied Logistic Regression* (3rd ed.). John Wiley & Sons, Inc.

Examples

```
# Example from Hosmer et al., 2013
# Applied Logistic Regression (3rd ed.), Chapter 5, Section 5.2.2

# Recode 'raterisk' into a binary variable 'raterisk_cat'
glow500 <- dplyr::mutate(
  glow500,
  raterisk_cat = dplyr::case_when(
    raterisk %in% c("Less", "Same") ~ "C1",
    raterisk == "Greater" ~ "C2"
  )
)

# Fit a multiple logistic regression model with interactions
model.int <- glm(
  fracture ~ age + height + priorfrac + momfrac + armassist +
    raterisk_cat + age * priorfrac + momfrac * armassist,
  family = binomial,
  data = glow500
)

# Apply the Stukels test for goodness-of-fit
stukels_test(model.int)
```

---

summarize_results *Summarize Bayesian Logistic Regression Model Results*

Description

This function provides a visual and numeric summary of a Bayesian logistic regression model fitted with `rstan`. It displays posterior distributions of selected parameters and performs posterior predictive checks based on either user-provided simulations or internal simulations.

Usage

```
summarize_results(
  model,
  ypredict = NULL,
  data,
  outcome,
  intercept = NULL,
  var.param,
  rounding = 2,
  prob = 0.8,
  point.est = "median"
)
```

Arguments

`model`	A fitted model object of class `stanfit`, from `rstan` or `stanreg`, from `rstanarm`, representing a single-level logistic regression.
`ypredict`	Optional. A matrix of posterior predictive simulations of the outcome variable (e.g., generated externally). If `NULL`, predictions will be simulated internally assuming a single-level logistic regression. The matrix should have dimensions `S x N`, where `S` is the number of posterior draws (rows) and `N` is the number of observations (columns).
`data`	A data frame containing the variables used in the model.
`outcome`	A character string specifying the name of the binary outcome variable in `data`.
`intercept`	Optional. A character string naming the intercept parameter in the model. Defaults to `NULL`.
`var.param`	A named character vector mapping dataset variable names to model parameter names. When `ypredict` is `NULL`, the variable names must exist in `data` for posterior prediction. If `ypredict` is provided, this check is skipped since predictions are supplied directly.
`rounding`	An integer specifying the number of decimal places for printed parameter summaries. Must be a non-negative integer. Default is 2.
`prob`	A numeric value between 0 and 1 specifying the width of the credible interval for posterior plots (e.g., 0.8 for 80% intervals). Default is 0.8.
`point.est`	Character string, either `"mean"` or `"median"`, indicating the point estimate to use in posterior plots. Default is median.

Value

A list with the following components:

posterior_plot A `ggplot` object showing posterior distributions of the coefficients.

ppc_mean A `ggplot` object showing the posterior predictive check for the point estimate.

ppc_sd A `ggplot` object showing the posterior predictive check for the standard deviation.

Also prints parameter summaries and displays plots.

`univariable.clogmodels`

Univariable Conditional Logistic Regression Models

Description

This function fits separate univariable conditional logistic regression models for a specified outcome and a list of explanatory variables, using the `clogit` function from the `survival` package. It returns a summary table with coefficients, standard errors, p-values, and optionally odds ratios with confidence intervals. The strata vector must be in the same order as the observations in the `data` frame.

Usage

```
univariable.clogmodels(
  data,
  yval,
  xval,
  strata,
  OR = FALSE,
  inc.or = NULL,
  confidence.level = 0.95
)
```

Arguments

`data`	A data frame containing the outcome, predictors, and strata variables
`yval`	A string indicating the name of the binary outcome variable.
`xval`	A character vector of predictor variable names for which univariable models will be fit.
`strata`	A character string specifying the name of the stratification variable (e.g., matching ID).
`OR`	Logical; if TRUE, odds ratios and their confidence intervals are returned. Default to FALSE.
`inc.or`	A numeric vector specifying the unit increment to apply when calculating odds ratios for each coefficient. Required if `OR = TRUE`. The length of `inc.or` must match the number of coefficients produced across all univariable models (including dummy variables for factor levels, if applicable). It must align with the order of `xval`: for each variable, include one value per resulting coefficient.
`confidence.level`	The confidence level to use for interval estimation of odds ratios. Defaults to 0.95.

Details

Each predictor in `xval` is fit in a separate model with the outcome and stratification variable. When `OR = TRUE`, the user must supply `inc.or`, a vector of increments matching the number of coefficients across all univariable models (i.e., including levels of factors if applicable). The function assumes that the outcome is binary and numeric (0/1); it will attempt to coerce it if not.

Value

A data frame with coefficients from the univariable models, standard errors, p-values, and, if requested, odds ratios with lower and upper confidence limits.

Examples

```
# Example from Hosmer et al., 2013
# Applied Logistic Regression (3rd ed.), Chapter 7, Table 7.1
```

```
 # Convert 'fracture' to binary (0 = No, 1 = Yes)
glow11m$fracture <- ifelse(glow11m$fracture == "Yes", 1, 0)

# Define variables to evaluate
unvariables <- c(
  "height", "weight", "bmi", "priorfrac", "premeno", "momfrac",
  "armassist", "smoke", "raterisk")

# Define value ranges used to interpret odds ratios (Optional)
val.pe <- c(10, 10, 3, 1, 1, 1, 1, 1, 1, 1)

# Run univariable conditional logistic regressions
univariable.clogmodels(glow11m, yval = 'fracture', xval = unvariables,
                        strata = 'pair', OR = TRUE, inc.or = val.pe)
```

univariable.models *Univariable Logistic Regression Summary Table*

Description

Fits univariable logistic regression models glm function for a set of predictors and summarizes coefficients, standard errors, likelihood ratio test statistics, and optionally odds ratios with confidence intervals.

Usage

```
univariable.models(
  data,
  yval,
  xval,
  OR = FALSE,
  inc.or = NULL,
  confidence.level = 0.95
)
```

Arguments

data	A data frame containing the outcome and predictor variables.
yval	A string indicating the name of the binary outcome variable.
xval	A character vector with the names of the explanatory variables to be evaluated univariately.
OR	Logical; if TRUE, odds ratios and their confidence intervals are returned. Default to FALSE.
inc.or	A numeric vector of scaling factors to be applied to each coefficient to obtain odds ratios.
confidence.level	The confidence level to use for interval estimation of odds ratios. Defaults to 0.95.

Value

A data frame with coefficients from the univariable models, standard errors, p-values, and, if requested, odds ratios with lower and upper confidence limits.

Examples

```
# Example from Hosmer et al., 2013
# Applied Logistic Regression (3rd ed.), Chapter 4, Table 4.7

# Define variables to evaluate
unvariables <- c(
  'age','weight','height', 'bmi', 'priorfrac', 'premeno', 'momfrac',
  'armassist','smoke', 'raterisk')

# Define value ranges used to interpret odds ratios (Optional)
val.pe <- c(5, 5, 10, 5, 1, 1, 1, 1, 1, 1, 1)

# Run univariable conditional logistic regressions
univariable.models(glow500, yval = 'fracture', xval = unvariables, OR = TRUE, inc.or = val.pe)
```

Index