Package ‘RprobitB’

February 26, 2024

Type Package

Title Bayesian Probit Choice Modeling

Version 1.1.4

Description Bayes estimation of probit choice models, both in the cross-sectional and panel setting. The package can analyze binary, multivariate, ordered, and ranked choices, as well as heterogeneity of choice behavior among deciders. The main functionality includes model fitting via Markov chain Monte Carlo methods, tools for convergence diagnostic, choice data simulation, in-sample and out-of-sample choice prediction, and model selection using information criteria and Bayes factors. The latent class model extension facilitates preference-based decider classification, where the number of latent classes can be inferred via the Dirichlet process or a weight-based updating heuristic. This allows for flexible modeling of choice behavior without the need to impose structural constraints. For a reference on the method see Oelschlaeger and Bauer (2021) [https://trid.trb.org/view/1759753].


BugReports https://github.com/loelschlaeger/RprobitB/issues

License GPL-3

Encoding UTF-8

Imports checkmate, cli, crayon, doSNOW, foreach, ggplot2, graphics, gridExtra, MASS, mixtools, mvtnorm, oeli (>= 0.4.1), parallel, plotROC, progress, Rcpp, Rdpack, rlang, stats, utils, viridis

LinkingTo Rcpp, RcppArmadillo

Suggests knitr, mlogit, testthat (>= 3.0.0)

Depends R (>= 3.5.0)

RoxygenNote 7.3.1

RdMacros Rdpack

VignetteBuilder knitr
LazyData: true
LazyDataCompression: xz
NeedsCompilation: yes

Author: Lennart Oelschläger [aut, cre]
        (https://orcid.org/0000-0001-5421-9313),
        Dietmar Bauer [aut] (https://orcid.org/0000-0003-2920-7032),
        Sebastian Büscher [ctb],
        Manuel Batram [ctb]

Maintainer: Lennart Oelschläger <oelschlaeger.lennart@gmail.com>

Repository: CRAN

Date/Publication: 2024-02-26 14:10:06 UTC

R topics documented:

as_cov_names .................................................. 3
check_form ..................................................... 3
check_prior ...................................................... 4
choice_probabilities ......................................... 6
classification .................................................. 7
coeff.RprobitB_fit ............................................ 8
compute_p_si ................................................... 9
cov_mix .......................................................... 9
create_lagged_cov ............................................. 10
fit_model ....................................................... 11
get_cov ........................................................ 13
mml .............................................................. 13
model_selection ............................................... 14
npar .............................................................. 15
overview_effects .............................................. 15
plot.RprobitB_data ........................................... 17
plot.RprobitB_fit ............................................. 17
plot_roc ........................................................ 18
point_estimates ............................................... 19
predict.RprobitB_fit ......................................... 19
pred_acc ....................................................... 21
prepare_data .................................................. 21
RprobitB_parameter ........................................... 24
R_hat ............................................................ 26
simulate_choices ............................................... 27
train_choice ................................................... 29
train_test ..................................................... 30
transform.RprobitB_fit ....................................... 31
update.RprobitB_fit .......................................... 32

Index: 36
as_cov_names

Re-label alternative specific covariates

Description

In \texttt{RprobitB}, alternative specific covariates must be named in the format "<covariate>_<alternative>". This convenience function generates the format for a given choice_data set.

Usage

\begin{verbatim}
as_cov_names(choice_data, cov, alternatives)
\end{verbatim}

Arguments

\begin{itemize}
\item choice_data A \texttt{data.frame} of choice data in wide format, i.e. each row represents one choice occasion.
\item cov A character vector of the names of alternative specific covariates in choice_data.
\item alternatives A (character or numeric) vector of the alternative names.
\end{itemize}

Value

The \texttt{choice_data} input with updated column names.

Examples

\begin{verbatim}
data("Electricity", package = "mlogit")
cov <- c("pf", "cl", "loc", "wk", "tod", "seas")
alternatives <- 1:4

colnames(Electricity)

Electricity <- as_cov_names(Electricity, cov, alternatives)

colnames(Electricity)
\end{verbatim}

check_form

Check model formula

Description

This function checks the input form.

Usage

\begin{verbatim}
check_form(form, re = NULL, ordered = FALSE)
\end{verbatim}
check_prior

Arguments

form A formula object that is used to specify the model equation. The structure is choice ~ A | B | C, where
- choice is the name of the dependent variable (the choices),
- A are names of alternative and choice situation specific covariates with a coefficient that is constant across alternatives,
- B are names of choice situation specific covariates with alternative specific coefficients,
- and C are names of alternative and choice situation specific covariates with alternative specific coefficients.

Multiple covariates (of one type) are separated by a + sign. By default, alternative specific constants (ASCs) are added to the model. They can be removed by adding +0 in the second spot.

In the ordered probit model (ordered = TRUE), the formula object has the simple structure choice ~ A. ASCs are not estimated.

re A character (vector) of covariates of form with random effects. If re = NULL (the default), there are no random effects. To have random effects for the ASCs, include "ASC" in re.

ordered A boolean, FALSE per default. If TRUE, the choice set alternatives is assumed to be ordered from worst to best.

Value

A list that contains the following elements:
- The input form.
- The name choice of the dependent variable in form.
- The input re.
- A list vars of three character vectors of covariate names of the three covariate types.
- A boolean ASC, determining whether the model has ASCs.

See Also

overview_effects() for an overview of the model effects

Description

This function checks the compatibility of submitted parameters for the prior distributions and sets missing values to default values.
check_prior

Usage

check_prior(
  P_f,
  P_r,
  J,
  ordered = FALSE,
  eta = numeric(P_f),
  Psi = diag(P_f),
  delta = 1,
  xi = numeric(P_r),
  D = diag(P_r),
  nu = P_r + 2,
  Theta = diag(P_r),
  kappa = if (ordered) 4 else (J + 1),
  E = if (ordered) diag(1) else diag(J - 1),
  zeta = numeric(J - 2),
  Z = diag(J - 2)
)

Arguments

P_f  The number of covariates connected to a fixed coefficient (can be 0).
P_r  The number of covariates connected to a random coefficient (can be 0).
J    The number (greater or equal 2) of choice alternatives.
ondered A boolean, FALSE per default. If TRUE, the choice set alternatives is assumed
to be ordered from worst to best.
etra The mean vector of length P_f of the normal prior for alpha. Per default, eta =
numeric(P_f).
Psi  The covariance matrix of dimension P_f x P_f of the normal prior for alpha.
Per default, Psi = diag(P_f).
delta A numeric for the concentration parameter vector rep(delta,C) of the Dirich-
let prior for s. Per default, delta = 1. In case of Dirichlet process-based updates
of the latent classes, delta = 0.1 per default.
xi   The mean vector of length P_r of the normal prior for each b_c. Per default, xi =
numeric(P_r).
D    The covariance matrix of dimension P_r x P_r of the normal prior for each b_c.
Per default, D = diag(P_r).
nu   The degrees of freedom (a natural number greater than P_r) of the Inverse
Wishart prior for each Omega_c. Per default, nu = P_r + 2.
Theta The scale matrix of dimension P_r x P_r of the Inverse Wishart prior for each
Omega_c. Per default, Theta = diag(P_r).
kappa The degrees of freedom (a natural number greater than J-1) of the Inverse
Wishart prior for Sigma. Per default, kappa = J + 1.
E    The scale matrix of dimension J-1 x J-1 of the Inverse Wishart prior for Sigma.
Per default, E = diag(J - 1).
choice_probabilities

zeta
The mean vector of length $J - 2$ of the normal prior for the logarithmic increments $d$ of the utility thresholds in the ordered probit model. Per default, $\zeta = \text{numeric}(J - 2)$.

$Z$
The covariance matrix of dimension $(J - 2) \times (J - 2)$ of the normal prior for the logarithmic increments $d$ of the utility thresholds in the ordered probit model. Per default, $Z = \text{diag}(J - 2)$.

Details
A priori, we assume that the model parameters follow these distributions:

- $\alpha \sim N(\eta, \Psi)$
- $s \sim \text{Dir}(\delta)$
- $b_c \sim N(\xi, D)$ for all classes $c$
- $\Omega_c \sim \text{IW}(\nu, \Theta)$ for all classes $c$
- $\Sigma \sim \text{IW}(\kappa, E)$
- $d \sim N(\zeta, Z)$

where $N$ denotes the normal, $\text{Dir}$ the Dirichlet, and $\text{IW}$ the Inverted Wishart distribution.

Value
An object of class RprobitB_prior, which is a list containing all prior parameters. Parameters that are not relevant for the model configuration are set to NA.

Examples
check_prior(P_f = 1, P_r = 2, J = 3, ordered = TRUE)

choice_probabilities  Compute choice probabilities

Description
This function returns the choice probabilities of an RprobitB_fit object.

Usage
choice_probabilities(x, data = NULL, par_set = mean)
classification

Arguments

x  
An object of class RprobitB_fit.

data  
Either NULL or an object of class RprobitB_data. In the former case, choice probabilities are computed for the data that was used for model fitting. Alternatively, a new data set can be provided.

par_set  
Specifying the parameter set for calculation and either

- a function that computes a posterior point estimate (the default is mean()),
- "true" to select the true parameter set,
- an object of class RprobitB_parameter.

Value

A data frame of choice probabilities with choice situations in rows and alternatives in columns. The first two columns are the decider identifier "id" and the choice situation identifier "idc".

Examples

data <- simulate_choices(form = choice ~ covariate, N = 10, T = 10, J = 2)
x <- fit_model(data)
choice_probabilities(x)

classification  Classify deciders preference-based

Description

This function classifies the deciders based on their allocation to the components of the mixing distribution.

Usage

classification(x, add_true = FALSE)

Arguments

x  
An object of class RprobitB_fit.

add_true  
Set to TRUE to add true class memberships to output (if available).
Details

The function can only be used if the model has at least one random effect (i.e. \( P_r >= 1 \)) and at least two latent classes (i.e. \( C >= 2 \)).

In that case, let \( z_1, \ldots, z_N \) denote the class allocations of the \( N \) deciders based on their estimated mixed coefficients \( \beta = (\beta_1, \ldots, \beta_N) \). Independently for each decider \( n \), the conditional probability \( \Pr(z_n = c | s, \beta_n, b, \Omega) \) of having \( \beta_n \) allocated to class \( c \) for \( c = 1, \ldots, C \) depends on the class allocation vector \( s \), the class means \( b = (b_c)_C \) and the class covariance matrices \( \Omega = (\Omega_c)_C \) and is proportional to

\[
s_c \phi(\beta_n | b_c, \Omega_c).
\]

This function displays the relative frequencies of which each decider was allocated to the classes during the Gibbs sampling. Only the thinned samples after the burn-in period are considered.

Value

A data frame. The row names are the decider ids. The first \( C \) columns contain the relative frequencies with which the deciders are allocated to the \( C \) classes. Next, the column \( \text{est} \) contains the estimated class of the decider based on the highest allocation frequency. If \( \text{add\_true} \), the next column \( \text{true} \) contains the true class memberships.

See Also

update_z() for the updating function of the class allocation vector.
**compute_p_si**

*Compute choice probabilities at posterior samples*

**Description**

This function computes the probability for each observed choice at the (normalized, burned and thinned) samples from the posterior. These probabilities are required to compute the WAIC and the marginal model likelihood mml.

**Usage**

```r
compute_p_si(x, ncores = parallel::detectCores() - 1, recompute = FALSE)
```

**Arguments**

- `x` An object of class RprobitB_fit.
- `ncores` This function is parallelized, set the number of cores here.
- `recompute` Set to TRUE to recompute the probabilities.

**Value**

The object `x`, including the object `p_si`, which is a matrix of probabilities, observations in rows and posterior samples in columns.

---

**cov_mix**

*Extract estimated covariance matrix of mixing distribution*

**Description**

This convenience function returns the estimated covariance matrix of the mixing distribution.

**Usage**

```r
cov_mix(x, cor = FALSE)
```

**Arguments**

- `x` An object of class RprobitB_fit.
- `cor` If TRUE, returns the correlation matrix instead.

**Value**

The estimated covariance matrix of the mixing distribution. In case of multiple classes, a list of matrices for each class.
**create_lagged_cov**  

*Create lagged choice covariates*

**Description**

This function creates lagged choice covariates from the `data.frame` `choice_data`, which is assumed to be sorted by the choice occasions.

**Usage**

```r
create_lagged_cov(choice_data, column, k = 1, id = "id")
```

**Arguments**

- `choice_data`  
  A `data.frame` of choice data in wide format, i.e. each row represents one choice occasion.

- `column`  
  A character, the column name in `choice_data`, i.e. the covariate name. Can be a vector.

- `k`  
  A positive number, the number of lags (in units of observations), see the details. Can be a vector. The default is `k = 1`.

- `id`  
  A character, the name of the column in `choice_data` that contains unique identifier for each decision maker. The default is "id".

**Details**

Say that `choice_data` contains the column `column`. Then, the function call

```r
create_lagged_cov(choice_data, column, k, id)
```

returns the input `choice_data` which includes a new column named `column.k`. This column contains for each decider (based on `id`) and each choice occasion the covariate faced before `k` choice occasions. If this data point is not available, it is set to `NA`. In particular, the first `k` values of `column.k` will be `NA` (initial condition problem).

**Value**

The input `choice_data` with the additional columns named `column.k` for each element `column` and each number `k` containing the lagged covariates.
**fit_model**  

*Fit probit model to choice data*

**Description**

This function performs Markov chain Monte Carlo simulation for fitting different types of probit models (binary, multivariate, mixed, latent class, ordered, ranked) to discrete choice data.

**Usage**

```r
fit_model(
  data,
  scale = "Sigma_1,1 := 1",
  R = 1000,
  B = R/2,
  Q = 1,
  print_progress =getOption("RprobitB_progress"),
  prior = NULL,
  latent_classes = NULL,
  seed = NULL,
  fixed_parameter = list()
)
```

**Arguments**

- **data**: An object of class `RprobitB_data`.
- **scale**: A character which determines the utility scale. It is of the form `<parameter> := <value>`, where `<parameter>` is either the name of a fixed effect or `Sigma_<j>,<j>`, `<j>` for the `<j>`th diagonal element of Sigma, and `<value>` is the value of the fixed parameter.
- **R**: The number of iterations of the Gibbs sampler.
- **B**: The length of the burn-in period, i.e. a non-negative number of samples to be discarded.
- **Q**: The thinning factor for the Gibbs samples, i.e. only every `Q`th sample is kept.
- **print_progress**: A boolean, determining whether to print the Gibbs sampler progress and the estimated remaining computation time.
- **prior**: A named list of parameters for the prior distributions. See the documentation of `check_prior` for details about which parameters can be specified.
- **latent_classes**: Either `NULL` (for no latent classes) or a list of parameters specifying the number of latent classes and their updating scheme:
  - **C**: The fixed number (greater or equal 1) of latent classes, which is set to 1 per default. If either `weight_update = TRUE` or `dp_update = TRUE` (i.e. if classes are updated), `C` equals the initial number of latent classes.
  - **weight_update**: A boolean, set to `TRUE` to weight-based update the latent classes. See ... for details.
• dp_update: A boolean, set to TRUE to update the latent classes based on a Dirichlet process. See ... for details.
• Cmax: The maximum number of latent classes.
• buffer: The number of iterations to wait before a next weight-based update of the latent classes.
• epsmin: The threshold weight (between 0 and 1) for removing a latent class in the weight-based updating scheme.
• epsmax: The threshold weight (between 0 and 1) for splitting a latent class in the weight-based updating scheme.
• distmin: The (non-negative) threshold in class mean difference for joining two latent classes in the weight-based updating scheme.

seed Set a seed for the Gibbs sampling.

fixed_parameter

Optionally specify a named list with fixed parameter values for alpha, C, s, b, Omega, Sigma, Sigma_full, beta, z, or d for the simulation. See the vignette on model definition for definitions of these variables.

Details

See the vignette on model fitting for more details.

Value

An object of class RprobitB_fit.

See Also

• prepare_data() and simulate_choices() for building an RprobitB_data object
• update() for estimating nested models
• transform() for transforming a fitted model

Examples

data <- simulate_choices(
  form = choice ~ var | 0, N = 100, T = 10, J = 3, seed = 1
)
model <- fit_model(data = data, R = 1000, seed = 1)
summary(model)
get_cov

**Extract covariates of choice occasion**

**Description**
This convenience function returns the covariates and the choices of specific choice occasions.

**Usage**

```r
get_cov(x, id, idc, idc_label)
```

**Arguments**

- `x`: Either an object of class `RprobitB_data` or `RprobitB_fit`.
- `id`: A numeric (vector), that specifies the decider(s).
- `idc`: A numeric (vector), that specifies the choice occasion(s).
- `idc_label`: The name of the column that contains the choice occasion identifier.

**Value**
A subset of the `choice_data` data frame specified in `prepare_data()`.

---

mml

**Approximate marginal model likelihood**

**Description**
This function approximates the model’s marginal likelihood.

**Usage**

```r
mml(x, S = 0, ncores = parallel::detectCores() - 1, recompute = FALSE)
```

**Arguments**

- `x`: An object of class `RprobitB_fit`.
- `S`: The number of prior samples for the prior arithmetic mean estimate. Per default, $S = 0$. In this case, only the posterior samples are used for the approximation via the posterior harmonic mean estimator, see the details section.
- `ncores`: Computation of the prior arithmetic mean estimate is parallelized, set the number of cores.
- `recompute`: Set to `TRUE` to recompute the likelihood.
Details

The model's marginal likelihood \( p(y \mid M) \) for a model \( M \) and data \( y \) is required for the computation of Bayes factors. In general, the term has no closed form and must be approximated numerically.

This function uses the posterior Gibbs samples to approximate the model's marginal likelihood via the posterior harmonic mean estimator. To check the convergence, call \( \text{plot(x$mml)} \), where \( x \) is the output of this function. If the estimation does not seem to have converged, you can improve the approximation by combining the value with the prior arithmetic mean estimator. The final approximation of the model's marginal likelihood than is a weighted sum of the posterior harmonic mean estimate and the prior arithmetic mean estimate, where the weights are determined by the sample sizes.

Value

The object \( x \), including the object \( \text{mml} \), which is the model's approximated marginal likelihood value.

---

**model_selection**

*Compare fitted models*

---

Description

This function returns a table with several criteria for model comparison.

Usage

```r
model_selection(
  ..., 
  criteria = c("npar", "LL", "AIC", "BIC"), 
  add_form = FALSE
)
```

Arguments

- `...`: One or more objects of class `RprobitB_fit`.
- `criteria`: A vector of one or more of the following characters:
  - "npar" for the number of model parameters (see \text{npar}),
  - "LL" for the log-likelihood value (see \text{logLik}),
  - "AIC" for the AIC value (see \text{AIC}),
  - "BIC" for the BIC value (see \text{BIC}),
  - "WAIC" for the WAIC value (also shows its standard error \text{sd(WAIC)} and the number \text{pWAIC} of effective model parameters, see \text{WAIC}),
  - "MMLL" for the marginal model log-likelihood,
  - "BF" for the Bayes factor,
  - "pred_acc" for the prediction accuracy (see \text{pred_acc}).
- `add_form`: Set to TRUE to add the model formulas.
Details

See the vignette on model selection for more details.

Value

A data frame, criteria in columns, models in rows.

---

npar

Extract number of model parameters

Description

This function extracts the number of model parameters of an RprobitB_fit object.

Usage

npar(object, ...)

## S3 method for class 'RprobitB_fit'

npar(object, ...)

Arguments

- object: An object of class RprobitB_fit.
- ...: Optionally more objects of class RprobitB_fit.

Value

Either a numeric value (if just one object is provided) or a numeric vector.

---

overview_effects

Print effect overview

Description

This function gives an overview of the effect names, whether the covariate is alternative-specific, whether the coefficient is alternative-specific, and whether it is a random effect.

Usage

overview_effects(
  form,
  re = NULL,
  alternatives,
  base = tail(alternatives, 1),
  ordered = FALSE
)
Arguments

form  A formula object that is used to specify the model equation. The structure is choice ~ A | B | C, where
       • choice is the name of the dependent variable (the choices),
       • A are names of alternative and choice situation specific covariates with a coefficient that is constant across alternatives,
       • B are names of choice situation specific covariates with alternative specific coefficients,
       • and C are names of alternative and choice situation specific covariates with alternative specific coefficients.

Multiple covariates (of one type) are separated by a + sign. By default, alternative specific constants (ASCs) are added to the model. They can be removed by adding +0 in the second spot.

In the ordered probit model (ordered = TRUE), the formula object has the simple structure choice ~ A. ASCs are not estimated.

re    A character (vector) of covariates of form with random effects. If re = NULL (the default), there are no random effects. To have random effects for the ASCs, include "ASC" in re.

alternatives A character vector with the names of the choice alternatives. If not specified, the choice set is defined by the observed choices. If ordered = TRUE, alternatives is assumed to be specified with the alternatives ordered from worst to best.

base   A character, the name of the base alternative for covariates that are not alternative specific (i.e. type 2 covariates and ASCs). Ignored and set to NULL if the model has no alternative specific covariates (e.g. in the ordered probit model). Per default, base is the last element of alternatives.

ordered A boolean, FALSE per default. If TRUE, the choice set alternatives is assumed to be ordered from worst to best.

Value

A data frame, each row is an effect, columns are the effect name "effect", and booleans whether the covariate is alternative-specific "as_value", whether the coefficient is alternative-specific "as_coef", and whether it is a random effect "random".

See Also

check_form() for checking the model formula specification.

Examples

overview_effects(
  form = choice ~ price + time + comfort + change | 1,
  re = c("price", "time"),
  alternatives = c("A", "B"),
  base = "A"
)
plot.RprobitB_data

### Description

This function is the plot method for an object of class `RprobitB_data`.

### Usage

```r
## S3 method for class 'RprobitB_data'
plot(x, by_choice = FALSE, alpha = 1, position = "dodge", ...)
```

### Arguments

- **x**: An object of class `RprobitB_data`.
- **by_choice**: Set to `TRUE` to group the covariates by the chosen alternatives.
- **alpha**, **position**: Passed to `ggplot`.
- **...**: Ignored.

### Value

No return value. Draws a plot to the current device.

### Examples

```r
data <- simulate_choices(
  form = choice ~ cost | 0,
  N = 100,
  T = 10,
  J = 2,
  alternatives = c("bus", "car"),
  true_parameter = list("alpha" = -1)
)
plot(data, by_choice = TRUE)
```

plot.RprobitB_fit

### Description

This function is the plot method for an object of class `RprobitB_fit`.

### Usage

```r
## S3 method for class 'RprobitB_fit'
plot(x, type = NULL, ignore = NULL, ...)
```
Argument

x  An object of class `RprobitB_fit`.

type  The type of plot, which can be one of:

  - "mixture" to visualize the mixing distribution,
  - "acf" for autocorrelation plots of the Gibbs samples,
  - "trace" for trace plots of the Gibbs samples,
  - "class_seq" to visualize the sequence of class numbers.

See the details section for visualization options.

ignore  A character (vector) of covariate or parameter names that do not get visualized.

...  Ignored.

Value

No return value. Draws a plot to the current device.

---

**plot_roc**  *Plot ROC curve*

**Description**

This function draws receiver operating characteristic (ROC) curves.

**Usage**

```r
plot_roc(..., reference = NULL)
```

**Arguments**

...  One or more `RprobitB_fit` objects or `data.frame`s of choice probability.

reference  The reference alternative.

**Value**

No return value. Draws a plot to the current device.
point_estimates  

**Compute point estimates**

**Description**
This function computes the point estimates of an RprobitB_fit. Per default, the mean of the Gibbs samples is used as a point estimate. However, any statistic that computes a single numeric value out of a vector of Gibbs samples can be specified for FUN.

**Usage**

```r
point_estimates(x, FUN = mean)
```

**Arguments**

- `x`: An object of class `RprobitB_fit`.
- `FUN`: A function that computes a single numeric value out of a vector of numeric values.

**Value**

An object of class `RprobitB_parameter`.

**Examples**

```r
data <- simulate_choices(form = choice ~ covariate, N = 10, T = 10, J = 2)
model <- fit_model(data)
point_estimates(model)
point_estimates(model, FUN = median)
```

---

predict.RprobitB_fit  

**Predict choices**

**Description**
This function predicts the discrete choice behavior.

**Usage**

```r
## S3 method for class 'RprobitB_fit'
predict(object, data = NULL, overview = TRUE, digits = 2, ...)
```
predict.RprobitB_fit

Arguments

object An object of class RprobitB_fit.
data Either
  • NULL, using the data in object,
  • an object of class RprobitB_data, for example the test part generated by train_test,
  • or a data frame of custom choice characteristics. It must have the same structure as choice_data used in prepare_data. Missing columns or NA values are set to 0.
overview If TRUE, returns a confusion matrix.
digits The number of digits of the returned choice probabilities. digits = 2 per default.
... Ignored.

details

Predictions are made based on the maximum predicted probability for each choice alternative. See the vignette on choice prediction for a demonstration on how to visualize the model’s sensitivity and specificity by means of a receiver operating characteristic (ROC) curve.

Value

Either a table if overview = TRUE or a data frame otherwise.

Examples

```r
data <- simulate_choices(
  form = choice ~ cov, N = 10, T = 10, J = 2, seed = 1
)
data <- train_test(data, test_proportion = 0.5)
model <- fit_model(data$train)

predict(model)
predict(model, overview = FALSE)
predict(model, data = data$test)
predict(
  model,
  data = data.frame("cov_A" = c(1, 1, NA, NA), "cov_B" = c(1, NA, 1, NA)),
  overview = FALSE
)
```
**pred_acc**

**Compute prediction accuracy**

**Description**

This function computes the prediction accuracy of an RprobitB_fit object. Prediction accuracy means the share of choices that are correctly predicted by the model, where prediction is based on the maximum choice probability.

**Usage**

```r
pred_acc(x, ...)
```

**Arguments**

- `x` An object of class RprobitB_fit.
- `...` Optionally specify more RprobitB_fit objects.

**Value**

A numeric.

---

**prepare_data**

**Prepare choice data for estimation**

**Description**

This function prepares choice data for estimation.

**Usage**

```r
prepare_data(
  form,
  choice_data,
  re = NULL,
  alternatives = NULL,
  ordered = FALSE,
  ranked = FALSE,
  base = NULL,
  id = "id",
  idc = NULL,
  standardize = NULL,
  impute = "complete_cases"
)
```
Arguments

form A formula object that is used to specify the model equation. The structure is choice ~ A | B | C, where
- choice is the name of the dependent variable (the choices),
- A are names of alternative and choice situation specific covariates with a coefficient that is constant across alternatives,
- B are names of choice situation specific covariates with alternative specific coefficients,
- and C are names of alternative and choice situation specific covariates with alternative specific coefficients.

Multiple covariates (of one type) are separated by a + sign. By default, alternative specific constants (ASCs) are added to the model. They can be removed by adding +0 in the second spot.

In the ordered probit model (ordered = TRUE), the formula object has the simple structure choice ~ A. ASCs are not estimated.

choice_data A data.frame of choice data in wide format, i.e. each row represents one choice occasion.
re A character (vector) of covariates of form with random effects. If re = NULL (the default), there are no random effects. To have random effects for the ASCs, include "ASC" in re.
alternatives A character vector with the names of the choice alternatives. If not specified, the choice set is defined by the observed choices. If ordered = TRUE, alternatives is assumed to be specified with the alternatives ordered from worst to best.
ordered A boolean, FALSE per default. If TRUE, the choice set alternatives is assumed to be ordered from worst to best.
ranked TBA
base A character, the name of the base alternative for covariates that are not alternative specific (i.e. type 2 covariates and ASCs). Ignored and set to NULL if the model has no alternative specific covariates (e.g. in the ordered probit model). Per default, base is the last element of alternatives.
id A character, the name of the column in choice_data that contains unique identifier for each decision maker. The default is "id".
idc A character, the name of the column in choice_data that contains unique identifier for each choice situation of each decision maker. Per default, these identifier are generated by the order of appearance.
standardize A character vector of names of covariates that get standardized. Covariates of type 1 or 3 have to be addressed by <covariate>_<alternative>. If standardize = "all", all covariates get standardized.
impute A character that specifies how to handle missing covariate entries in choice_data, one of:
- "complete_cases", removes all rows containing missing covariate entries (the default),
prepare_data

- "zero", replaces missing covariate entries by zero (only for numeric columns),
- "mean", imputes missing covariate entries by the mean (only for numeric columns).

Details

Requirements for the data.frame choice_data:

- It **must** contain a column named id which contains unique identifier for each decision maker.
- It **can** contain a column named idc which contains unique identifier for each choice situation of each decision maker. If this information is missing, these identifier are generated automatically by the appearance of the choices in the data set.
- It **can** contain a column named choice with the observed choices, where choice must match the name of the dependent variable in form. Such a column is required for model fitting but not for prediction.
- It **must** contain a numeric column named \( p_j \) for each alternative specific covariate \( p \) in form and each choice alternative \( j \) in alternatives.
- It **must** contain a numeric column named \( q \) for each covariate \( q \) in form that is constant across alternatives.

In the ordered case (ordered = TRUE), the column choice must contain the full ranking of the alternatives in each choice occasion as a character, where the alternatives are separated by commas, see the examples.

See the vignette on choice data for more details.

Value

An object of class `RprobitB_data`.

See Also

- `check_form()` for checking the model formula
- `overview_effects()` for an overview of the model effects
- `create_lagged_cov()` for creating lagged covariates
- `as_cov_names()` for re-labeling alternative-specific covariates
- `simulate_choices()` for simulating choice data
- `train_test()` for splitting choice data into a train and test subset

Examples

```r
data <- prepare_data(
  form = choice ~ price + time + comfort + change | 0,
  choice_data = train_choice,
  re = c("price", "time"),
  id = "deciderID",
  idc = "occasionID",
  standardize = c("price", "time")
)`
RprobitB_parameter

Define probit model parameter

Description

This function creates an object of class `RprobitB_parameter`, which contains the parameters of a probit model. If `sample = TRUE`, missing parameters are sampled. All parameters are checked against the values of `P_f`, `P_r`, `J`, and `N`.

Usage

```r
RprobitB_parameter(
P_f,
P_r,
J,
N,
ordered = FALSE,
alpha = NULL,
C = NULL,
s = NULL,
b = NULL,
Omega = NULL,
Sigma = NULL,
Sigma_full = NULL,
beta = NULL,
z = NULL,
d = NULL,
seed = NULL,
sample = TRUE
)
```

Arguments

- `P_f`:
  The number of covariates connected to a fixed coefficient (can be 0).
- `P_r`:
  The number of covariates connected to a random coefficient (can be 0).
RprobitB_parameter

J  The number (greater or equal 2) of choice alternatives.
N  The number (greater or equal 1) of decision makers.
ordered A boolean, FALSE per default. If TRUE, the choice set alternatives is assumed to be ordered from worst to best.
alpha  The fixed coefficient vector of length \( P_f \). Set to NA if \( P_f = 0 \).
C  The number (greater or equal 1) of latent classes of decision makers. Set to NA if \( P_r = 0 \). Otherwise, \( C = 1 \) per default.
s  The vector of class weights of length \( C \). Set to NA if \( P_r = 0 \). For identifiability, the vector must be non-ascending.
b  The matrix of class means as columns of dimension \( P_r \times C \). Set to NA if \( P_r = 0 \).
Omega  The matrix of class covariance matrices as columns of dimension \( P_r \times P_r \times C \). Set to NA if \( P_r = 0 \).
Sigma  The differenced error term covariance matrix of dimension \( J-1 \times J-1 \) with respect to alternative \( J \). In case of ordered = TRUE, a numeric, the single error term variance.
Sigma_full  The error term covariance matrix of dimension \( J \times J \). Internally, \( \text{Sigma}_\text{full} \) gets differenced with respect to alternative \( J \), so it becomes an identified covariance matrix of dimension \( J-1 \times J-1 \). \( \text{Sigma}_\text{full} \) is ignored if \( \text{Sigma} \) is specified or ordered = TRUE.
beta  The matrix of the decision-maker specific coefficient vectors of dimension \( P_r \times N \). Set to NA if \( P_r = 0 \).
z  The vector of the allocation variables of length \( N \). Set to NA if \( P_r = 0 \).
d  The numeric vector of the logarithmic increases of the utility thresholds in the ordered probit case (ordered = TRUE) of length \( J-2 \).
seed  Set a seed for the sampling of missing parameters.
sample  A boolean, if TRUE (default) missing parameters get sampled.

Value

An object of class RprobitB_parameter, i.e. a named list with the model parameters alpha, C, s, b, Omega, Sigma, Sigma_full, beta, and z.

Examples

RprobitB_parameter(P_f = 1, P_r = 2, J = 3, N = 10)
R_hat

*Compute Gelman-Rubin statistic*

**Description**

This function computes the Gelman-Rubin statistic $R_{\text{hat}}$.

**Usage**

```
R_hat(samples, parts = 2)
```

**Arguments**

- `samples`: A vector or a matrix of samples from a Markov chain, e.g. Gibbs samples. If `samples` is a matrix, each column gives the samples for a separate run.
- `parts`: The number of parts to divide each chain into sub-chains.

**Value**

A numeric value, the Gelman-Rubin statistic.

**References**

[https://bookdown.org/rdpeng/advstatcomp/monitoring-convergence.html](https://bookdown.org/rdpeng/advstatcomp/monitoring-convergence.html)

**Examples**

```r
no_chains <- 2
length_chains <- 1e3
samples <- matrix(NA_real_, length_chains, no_chains)
samples[1, ] <- 1
Gamma <- matrix(c(0.8, 0.1, 0.2, 0.9), 2, 2)
for (c in 1:no_chains) {
  for (t in 2:length_chains) {
    samples[t, c] <- sample(1:2, 1, prob = Gamma[samples[t - 1, c], ])
  }
}
R_hat(samples)
```
**simulate_choices**  
*Simulate choice data*

**Description**  
This function simulates choice data from a probit model.

**Usage**

```r
simulate_choices(
  form,
  N,
  T = 1,
  J,
  re = NULL,
  alternatives = NULL,
  ordered = FALSE,
  ranked = FALSE,
  base = NULL,
  covariates = NULL,
  seed = NULL,
  true_parameter = list()
)
```

**Arguments**

- `form`  
  A formula object that is used to specify the model equation. The structure is `choice ~ A | B | C`, where
  - `choice` is the name of the dependent variable (the choices),
  - `A` are names of alternative and choice situation specific covariates with a coefficient that is constant across alternatives,
  - `B` are names of choice situation specific covariates with alternative specific coefficients,
  - and `C` are names of alternative and choice situation specific covariates with alternative specific coefficients.

  Multiple covariates (of one type) are separated by a `+` sign. By default, alternative specific constants (ASCs) are added to the model. They can be removed by adding `+0` in the second spot.

  In the ordered probit model (`ordered = TRUE`), the `formula` object has the simple structure `choice ~ A`. ASCs are not estimated.

- `N`  
  The number (greater or equal 1) of decision makers.

- `T`  
  The number (greater or equal 1) of choice occasions or a vector of choice occasions of length `N` (i.e. a decision maker specific number). Per default, `T = 1`.

- `J`  
  The number (greater or equal 2) of choice alternatives.
`simulate_choices` is a function that simulates choice data from a latent class model. The function takes several arguments:

- `form`: A character (vector) of covariates of `form` with random effects. If `re = NULL` (the default), there are no random effects. To have random effects for the ASCs, include "ASC" in `re`.
- `alternatives`: A character vector with the names of the choice alternatives. If not specified, the choice set is defined by the observed choices. If `ordered = TRUE`, alternatives is assumed to be specified with the alternatives ordered from worst to best.
- `ordered`: A boolean, `FALSE` per default. If `TRUE`, the choice set `alternatives` is assumed to be ordered from worst to best.
- `ranked`: TBA
- `base`: A character, the name of the base alternative for covariates that are not alternative specific (i.e. type 2 covariates and ASCs). Ignored and set to `NULL` if the model has no alternative specific covariates (e.g. in the ordered probit model). Per default, `base` is the last element of `alternatives`.
- `covariates`: A named list of covariate values. Each element must be a vector of length equal to the number of choice occasions and named according to a covariate. Covariates for which no values are supplied are drawn from a standard normal distribution.
- `seed`: Set a seed for the simulation.
- `true_parameter`: Optionally specify a named list with true parameter values for `alpha`, `C`, `s`, `b`, `Omega`, `Sigma`, `Sigma_full`, `beta`, `z`, or `d` for the simulation. See the vignette on model definition for definitions of these variables.

Details

See the vignette on choice data for more details.

Value

An object of class `RprobitB_data`.

See Also

- `check_form()` for checking the model formula
- `overview_effects()` for an overview of the model effects
- `create_lagged_cov()` for creating lagged covariates
- `as_cov_names()` for re-labeling alternative-specific covariates
- `prepare_data()` for preparing empirical choice data
- `train_test()` for splitting choice data into a train and test subset

Examples

```r
### simulate data from a binary probit model with two latent classes
data <- simulate_choices(
  form = choice ~ cost | income | time,
  N = 100,
```
train_choice

Stated Preferences for Train Traveling

Description

Data set of 2929 stated choices by 235 Dutch individuals deciding between two virtual train trip options "A" and "B" based on the price, the travel time, the number of rail-to-rail transfers (changes), and the level of comfort.

The data were obtained in 1987 by Hague Consulting Group for the National Dutch Railways. Prices were recorded in Dutch guilder and in this data set transformed to Euro at an exchange rate of 2.20371 guilders = 1 Euro.
train_test

Usage

train_choice

Format

A data.frame with 2929 rows and 11 columns:

- **deciderID**: integer identifier for the decider
- **occasionID**: integer identifier for the choice occasion
- **choice**: character for the chosen alternative (either "A" or "B")
- **price_A**: numeric price for alternative "A" in Euro
- **time_A**: numeric travel time for alternative "A" in hours
- **change_A**: integer number of changes for alternative "A"
- **comfort_A**: integer comfort level (in decreasing comfort order) for alternative "A"
- **price_B**: numeric price for alternative "B" in Euro
- **time_B**: numeric travel time for alternative "B" in hours
- **change_B**: integer number of changes for alternative "B"
- **comfort_B**: integer comfort level (in decreasing comfort order) for alternative "B"

References


train_test

Split choice data in train and test subset

Description

This function splits choice data into a train and a test part.

Usage

```r
train_test(
  x,
  test_proportion = NULL,
  test_number = NULL,
  by = "N",
  random = FALSE,
  seed = NULL
)
```
Arguments

- **x**: An object of class `RprobitB_data`.
- **test_proportion**: A number between 0 and 1, the proportion of the test subsample.
- **test_number**: A positive integer, the number of observations in the test subsample.
- **by**: One of "N" (split by deciders) and "T" (split by choice occasions).
- **random**: If TRUE, the subsamples are built randomly.
- **seed**: Set a seed for building the subsamples randomly.

Details

See the vignette on choice data for more details.

Value

A list with two objects of class `RprobitB_data`, named "train" and "test".

Examples

### simulate choices for demonstration
```r
x <- simulate_choices(form = choice ~ covariate, N = 10, T = 10, J = 2)
```

### 70% of deciders in the train subsample,
### 30% of deciders in the test subsample
```r
train_test(x, test_proportion = 0.3, by = "N")
```

### 2 randomly chosen choice occasions per decider in the test subsample,
### the rest in the train subsample
```r
train_test(x, test_number = 2, by = "T", random = TRUE, seed = 1)
```
Usage

```r
## S3 method for class 'RprobitB_fit'
transform(
  `.data`,
  B = NULL,
  Q = NULL,
  scale = NULL,
  check_preference_flip = TRUE,
  ...
)
```

Arguments

- `_data_` An object of class `RprobitB_fit`.
- `B` The length of the burn-in period, i.e. a non-negative number of samples to be discarded.
- `Q` The thinning factor for the Gibbs samples, i.e. only every `Q`th sample is kept.
- `scale` A character which determines the utility scale. It is of the form `<parameter> := <value>`, where `<parameter>` is either the name of a fixed effect or `Sigma_<j>,<j>` for the `<j>`th diagonal element of `Sigma`, and `<value>` is the value of the fixed parameter.
- `check_preference_flip` Set to `TRUE` to check for flip in preferences after new scale.
- `...` Ignored.

Details

See the vignette "Model fitting" for more details: `vignette("v03_model_fitting", package = "RprobitB")`.

Value

The transformed `RprobitB_fit` object.

Description

This function estimates a nested probit model based on a given `RprobitB_fit` object.
Usage

```r
## S3 method for class 'RprobitB_fit'
update(
  object,
  form,
  re,
  alternatives,
  id,
  idc,
  standardize,
  impute,
  scale,
  R,
  B,
  Q,
  print_progress,
  prior,
  latent_classes,
  seed,
  ...
)
```

Arguments

- **object**: An object of class `RprobitB_fit`.
- **form**: A formula object that is used to specify the model equation. The structure is `choice ~ A | B | C`, where
  - `choice` is the name of the dependent variable (the choices),
  - `A` are names of alternative and choice situation specific covariates with a coefficient that is constant across alternatives,
  - `B` are names of choice situation specific covariates with alternative specific coefficients,
  - `C` are names of alternative and choice situation specific covariates with alternative specific coefficients.
  
  Multiple covariates (of one type) are separated by a `+` sign. By default, alternative specific constants (ASCs) are added to the model. They can be removed by adding `+0` in the second spot.
  
  In the ordered probit model (ordered = TRUE), the `formula` object has the simple structure `choice ~ A`. ASCs are not estimated.

- **re**: A character (vector) of covariates of `form` with random effects. If `re = NULL` (the default), there are no random effects. To have random effects for the ASCs, include "ASC" in `re`.

- **alternatives**: A character vector with the names of the choice alternatives. If not specified, the choice set is defined by the observed choices. If `ordered = TRUE`, `alternatives` is assumed to be specified with the alternatives ordered from worst to best.
id  A character, the name of the column in choice_data that contains unique identifier for each decision maker. The default is "id".

idc  A character, the name of the column in choice_data that contains unique identifier for each choice situation of each decision maker. Per default, these identifier are generated by the order of appearance.

standardize  A character vector of names of covariates that get standardized. Covariates of type 1 or 3 have to be addressed by <covariate>_<alternative>. If standardize = "all", all covariates get standardized.

impute  A character that specifies how to handle missing covariate entries in choice_data, one of:
- "complete_cases", removes all rows containing missing covariate entries (the default),
- "zero", replaces missing covariate entries by zero (only for numeric columns),
- "mean", imputes missing covariate entries by the mean (only for numeric columns).

scale  A character which determines the utility scale. It is of the form <parameter> := <value>, where <parameter> is either the name of a fixed effect or Sigma_<j>,<j> for the <j>th diagonal element of Sigma, and <value> is the value of the fixed parameter.

R  The number of iterations of the Gibbs sampler.

B  The length of the burn-in period, i.e. a non-negative number of samples to be discarded.

Q  The thinning factor for the Gibbs samples, i.e. only every Qth sample is kept.

print_progress  A boolean, determining whether to print the Gibbs sampler progress and the estimated remaining computation time.

prior  A named list of parameters for the prior distributions. See the documentation of check_prior for details about which parameters can be specified.

latent_classes  Either NULL (for no latent classes) or a list of parameters specifying the number of latent classes and their updating scheme:
- C: The fixed number (greater or equal 1) of latent classes, which is set to 1 per default. If either weight_update = TRUE or dp_update = TRUE (i.e. if classes are updated), C equals the initial number of latent classes.
- weight_update: A boolean, set to TRUE to weight-based update the latent classes. See ... for details.
- dp_update: A boolean, set to TRUE to update the latent classes based on a Dirichlet process. See ... for details.
- Cmax: The maximum number of latent classes.
- buffer: The number of iterations to wait before a next weight-based update of the latent classes.
- epsmin: The threshold weight (between 0 and 1) for removing a latent class in the weight-based updating scheme.
- epsmax: The threshold weight (between 0 and 1) for splitting a latent class in the weight-based updating scheme.
- `distmin`: The (non-negative) threshold in class mean difference for joining two latent classes in the weight-based updating scheme.

- `seed`: Set a seed for the Gibbs sampling.

- `...`: Ignored.

**Details**

All parameters (except for `object`) are optional and if not specified retrieved from the specification for `object`.

**Value**

An object of class `RprobitB_fit`. 
Index

- dataset
  - train_choice, 29
AIC, 14
as_cov_names, 3
as_cov_names(), 23, 28
BIC, 14
check_form, 3
check_form(), 16, 23, 28
check_prior, 4, 11, 34
choice_probabilities, 6
classification, 7
coeff.RprobitB_fit, 8
compute_p_si, 9
cov_mix, 9
create_lagged_cov, 10
create_lagged_cov(), 23, 28
fit_model, 11
get_cov, 13
ggplot, 17
logLik, 14
mml, 9, 13
model_selection, 14
npar, 14, 15
overview_effects, 15
overview_effects(), 4, 23, 28
plot.RprobitB_data, 17
plot.RprobitB_fit, 17
plot_roc, 18
point_estimates, 19
pred_acc, 14, 21
predict.RprobitB_fit, 19
prepare_data, 20, 21
prepare_data(), 12, 28
R_hat, 26
RprobitB_fit, 18, 19, 32
RprobitB_parameter, 19, 24
simulate_choices, 27
simulate_choices(), 12, 23
train_choice, 29
train_test, 20, 30
train_test(), 23, 28
transform(), 12
transform.RprobitB_fit, 31
update(), 12
update.RprobitB_fit, 32
update_z(), 8
WAIC, 9, 14