

Package: choicer (via r-universe)

May 20, 2026

Title Discrete Choice Models for Economic Applications

Version 0.1.0

Description Fast estimation of discrete-choice models for applied economics. Likelihoods, analytical gradients and Hessians are implemented in C++ with 'OpenMP' parallelism, scaling efficiently to specifications with many alternative-specific constants. Post-estimation routines return predicted shares, own- and cross-price elasticities, and diversion ratios. Supports multinomial logit ('MNL'), mixed logit ('MXL'), and nested logit ('NL').

License LGPL (>= 3)

URL <https://github.com/fpcordeiro/choicer>

BugReports <https://github.com/fpcordeiro/choicer/issues>

Encoding UTF-8

Depends R (>= 4.1.0)

LinkingTo Rcpp, RcppArmadillo

Imports data.table, nloptr, randtoolbox, Rcpp, stats

Suggests testthat (>= 3.0.0), numDeriv, future.apply, goftest

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

NeedsCompilation yes

Author Fernando Cordeiro [aut, cre, cph]

Maintainer Fernando Cordeiro <fernandolpcordeiro@gmail.com>

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

Date/Publication 2026-05-20 09:10:07 UTC

RemoteUrl <https://github.com/cran/choicer>

RemoteRef HEAD

RemoteSha 7105dd7b4b34a24b5ae3fc5df92dd00db2917547

Contents

blp	3
blp.choicer_mnl	4
blp.choicer_mxl	5
blp_contraction	6
build_var_mat	7
coef.choicer_fit	8
diversion_ratios	8
diversion_ratios.choicer_mnl	9
diversion_ratios.choicer_mxl	10
elasticities	11
elasticities.choicer_mnl	12
elasticities.choicer_mxl	12
get_halton_normals	13
jacobian_vech_Sigma	14
logLik.choicer_fit	15
mc_asymptotics	15
mnl_diversion_ratios_parallel	17
mnl_elasticities_parallel	18
mnl_loglik_gradient_parallel	19
mnl_loglik_hessian_parallel	21
mnl_predict	22
mnl_predict_shares	23
monte_carlo	24
mxl_bhhh_parallel	25
mxl_blp_contraction	27
mxl_diversion_ratios_parallel	29
mxl_elasticities_parallel	30
mxl_hessian_parallel	32
mxl_loglik_gradient_parallel	33
mxl_predict	35
mxl_predict_shares	36
new_choicer_sim	37
nl_loglik_gradient_parallel	38
nl_loglik_numeric_hessian	39
nobs.choicer_fit	40
predict.choicer_mnl	41
predict.choicer_mxl	42
prepare_mnl_data	43
prepare_mxl_data	44
prepare_nl_data	46
print.choicer_fit	47
print.summary.choicer_mnl	48
print.summary.choicer_mxl	48
print.summary.choicer_nl	49
recovery_table	50
run_mnlogit	51

run_mxlogit	53
run_nestlogit	56
simulate_mnl_data	58
simulate_mxl_data	59
simulate_nl_data	61
summary.choicer_mnl	62
summary.choicer_mxl	62
summary.choicer_nl	63
vcov.choicer_fit	64

Index	65
--------------	-----------

blp	<i>BLP contraction mapping</i>
-----	--------------------------------

Description

Finds the ASC (delta) parameters such that predicted market shares match target shares, using the contraction mapping of Berry, Levinsohn, and Pakes (1995) [doi:10.2307/2171802](https://doi.org/10.2307/2171802).

Usage

```
blp(object, target_shares, ...)
```

Arguments

object	A fitted model object.
target_shares	Numeric vector of target market shares (length J).
...	Additional arguments passed to methods.

Value

Converged delta (ASC) vector.

Examples

```
library(data.table)
set.seed(42)
N <- 50; J <- 3
dt <- data.table(id = rep(1:N, each = J), alt = rep(1:J, N))
dt[, `:=`(x1 = rnorm(.N), x2 = rnorm(.N))]
dt[, choice := 0L]
dt[, choice := sample(c(1L, rep(0L, J - 1))), by = id]
fit <- run_mnlogit(dt, "id", "alt", "choice", c("x1", "x2"))
blp(fit, target_shares = rep(1/J, J))
```

blp.choicer_mnl *BLP contraction mapping for multinomial logit model*

Description

BLP contraction mapping for multinomial logit model

Usage

```
## S3 method for class 'choicer_mnl'
blp(
  object,
  target_shares,
  delta_init = NULL,
  tol = 1e-08,
  max_iter = 1000,
  ...
)
```

Arguments

object	A choicer_mnl object fitted with keep_data = TRUE.
target_shares	Numeric vector of target market shares. Length J_inside when no outside option, or J_inside + 1 (with the outside option's share at index 1) when include_outside_option = TRUE.
delta_init	Initial guess for delta (ASC) values. If NULL, uses the estimated ASCs from the fitted model.
tol	Convergence tolerance (default 1e-8).
max_iter	Maximum iterations (default 1000).
...	Additional arguments (ignored).

Value

Converged delta (ASC) vector.

Examples

```
library(data.table)
set.seed(42)
N <- 50; J <- 3
dt <- data.table(id = rep(1:N, each = J), alt = rep(1:J, N))
dt[, `:=`(x1 = rnorm(.N), x2 = rnorm(.N))]
dt[, choice := 0L]
dt[, choice := sample(c(1L, rep(0L, J - 1))), by = id]
fit <- run_mnlogit(dt, "id", "alt", "choice", c("x1", "x2"))
blp(fit, target_shares = rep(1/J, J))
```

blp.choicer_mxl *BLP contraction mapping for mixed logit model*

Description

BLP contraction mapping for mixed logit model

Usage

```
## S3 method for class 'choicer_mxl'
blp(
  object,
  target_shares,
  delta_init = NULL,
  tol = 1e-08,
  max_iter = 1000,
  ...
)
```

Arguments

object	A choicer_mxl object fitted with keep_data = TRUE.
target_shares	Numeric vector of target market shares. Length J_inside when no outside option, or J_inside + 1 (with the outside option's share at index 1) when include_outside_option = TRUE.
delta_init	Initial guess for delta (ASC) values. If NULL, uses the estimated ASCs from the fitted model.
tol	Convergence tolerance (default 1e-8).
max_iter	Maximum iterations (default 1000).
...	Additional arguments (ignored).

Value

Converged delta (ASC) vector.

Examples

```
library(data.table)
set.seed(42)
N <- 50; J <- 3
dt <- data.table(id = rep(1:N, each = J), alt = rep(1:J, N))
dt[, `:=`(x1 = rnorm(.N), w1 = rnorm(.N))]
dt[, choice := 0L]
dt[, choice := sample(c(1L, rep(0L, J - 1))), by = id]
fit <- run_mxlogit(
  data = dt, id_col = "id", alt_col = "alt", choice_col = "choice",
  covariate_cols = "x1", random_var_cols = "w1", S = 50L)
```

```
)
blp(fit, target_shares = rep(1/J, J))
```

blp_contraction	<i>BLP95 contraction mapping to find delta given target shares</i>
-----------------	--

Description

BLP95 contraction mapping to find delta given target shares

Usage

```
blp_contraction(
  delta,
  target_shares,
  X,
  beta,
  alt_idx,
  M,
  weights,
  include_outside_option = FALSE,
  tol = 1e-08,
  max_iter = 1000L
)
```

Arguments

delta	J x 1 vector with initial guess for deltas (ASCs)
target_shares	J x 1 vector with target shares for each alternative
X	sum(M) x K design matrix with covariates. M[i] x K matrix for individual i
beta	K x 1 vector with model parameters
alt_idx	sum(M) x 1 vector with indices of alternatives within each choice set; 1-based indexing
M	N x 1 vector with number of alternatives for each individual
weights	N x 1 vector with weights for each observation
include_outside_option	whether to include outside option normalized to 0 (if so, the outside option is not included in the data)
tol	convergence tolerance
max_iter	maximum number of iterations

Value

vector with contraction's delta (ASCs) output

Examples

```

library(data.table)
set.seed(42)
N <- 50; J <- 3
dt <- data.table(id = rep(1:N, each = J), alt = rep(1:J, N))
dt[, `:=`(x1 = rnorm(.N), x2 = rnorm(.N))]
dt[, choice := 0L]
dt[, choice := sample(c(1L, rep(0L, J - 1))), by = id]
fit <- run_mnlogit(dt, "id", "alt", "choice", c("x1", "x2"))
beta <- coef(fit)[fit$param_map$beta]
delta <- blp_contraction(rep(0, J), rep(1/J, J), fit$data$X,
  beta, fit$data$alt_idx, fit$data$M, fit$data$weights)
delta

```

 build_var_mat

 Reconstruct variance matrix L from L_params

Description

Reconstruct variance matrix L from L_params

Usage

```
build_var_mat(L_params, K_w, rc_correlation)
```

Arguments

L_params flattened choleski decomposition version of the random coefficient parameters matrix

K_w dimension of the random coefficient parameter (symmetric) matrix

rc_correlation whether random coefficients are correlated

Value

matrix equal to LL' , where L is the choleski decomposition of random coefficient matrix

Examples

```

L_params <- c(log(1.0), 0.3, log(0.5))
Sigma <- build_var_mat(L_params, K_w = 2, rc_correlation = TRUE)
Sigma # 2x2 covariance matrix

```

<code>coef.choicer_fit</code>	<i>Extract coefficients from a choicer_fit object</i>
-------------------------------	---

Description

Extract coefficients from a `choicer_fit` object

Usage

```
## S3 method for class 'choicer_fit'
coef(object, ...)
```

Arguments

<code>object</code>	A <code>choicer_fit</code> object.
<code>...</code>	Additional arguments (ignored).

Value

Named numeric vector of estimated coefficients.

Examples

```
library(data.table)
set.seed(42)
N <- 50; J <- 3
dt <- data.table(id = rep(1:N, each = J), alt = rep(1:J, N))
dt[, `:=`(x1 = rnorm(.N), x2 = rnorm(.N))]
dt[, choice := 0L]
dt[, choice := sample(c(1L, rep(0L, J - 1))), by = id]
fit <- run_mnlogit(dt, "id", "alt", "choice", c("x1", "x2"))
coef(fit)
```

<code>diversion_ratios</code>	<i>Compute aggregate diversion ratios</i>
-------------------------------	---

Description

Computes a $J \times J$ matrix of diversion ratios. Entry (i, j) is the fraction of demand lost by alternative j that is captured by alternative i when alternative j becomes less attractive.

Usage

```
diversion_ratios(object, ...)
```

Arguments

object A fitted model object.
 ... Additional arguments passed to methods.

Value

A J x J diversion ratio matrix with alternative labels.

Examples

```
library(data.table)
set.seed(42)
N <- 50; J <- 3
dt <- data.table(id = rep(1:N, each = J), alt = rep(1:J, N))
dt[, `:=`(x1 = rnorm(.N), x2 = rnorm(.N))]
dt[, choice := 0L]
dt[, choice := sample(c(1L, rep(0L, J - 1))), by = id]
fit <- run_mnlogit(dt, "id", "alt", "choice", c("x1", "x2"))
diversion_ratios(fit)
```

diversion_ratios.choicer_mnl

Diversion ratios for multinomial logit model

Description

Diversion ratios for multinomial logit model

Usage

```
## S3 method for class 'choicer_mnl'
diversion_ratios(object, ...)
```

Arguments

object A choicer_mnl object fitted with keep_data = TRUE.
 ... Additional arguments (ignored).

Value

A J x J diversion ratio matrix with alternative labels.

Examples

```

library(data.table)
set.seed(42)
N <- 50; J <- 3
dt <- data.table(id = rep(1:N, each = J), alt = rep(1:J, N))
dt[, `:=`(x1 = rnorm(.N), x2 = rnorm(.N))]
dt[, choice := 0L]
dt[, choice := sample(c(1L, rep(0L, J - 1))), by = id]
fit <- run_mnlogit(dt, "id", "alt", "choice", c("x1", "x2"))
diversion_ratios(fit)

```

diversion_ratios.choicer_mxl

Diversion ratios for mixed logit model

Description

Computes the attribute-based diversion ratio matrix. Entry (k, j) is the fraction of demand lost by alternative j that is captured by alternative k when a marginal change in alternative j's wrt_var attribute reduces s_j.

Usage

```

## S3 method for class 'choicer_mxl'
diversion_ratios(object, wrt_var, is_random_coef = FALSE, ...)

```

Arguments

object	A choicer_mxl object fitted with keep_data = TRUE.
wrt_var	Variable used to perturb alternative j's utility: a column name (character) or 1-based index. Indexes into X columns for fixed coefficients, or W columns for random coefficients (when is_random_coef = TRUE).
is_random_coef	Logical. TRUE if the variable has a random coefficient (is in W), FALSE if fixed (in X). Default FALSE.
...	Additional arguments (ignored).

Details

Unlike MNL, the MXL diversion ratio depends on which variable is perturbed: the realised coefficient β_{ik}^s varies across individuals and draws and does not cancel in the ratio. For a variable with a fixed coefficient the result is independent of the variable (β cancels); for a random-coefficient variable it is not.

Value

A J x J diversion ratio matrix with alternative labels. Cross-products are averaged across simulation draws inside the integration to avoid Jensen-style bias.

Examples

```

library(data.table)
set.seed(42)
N <- 50; J <- 3
dt <- data.table(id = rep(1:N, each = J), alt = rep(1:J, N))
dt[, `:=`(x1 = rnorm(.N), w1 = rnorm(.N))]
dt[, choice := 0L]
dt[, choice := sample(c(1L, rep(0L, J - 1))), by = id]
fit <- run_mxlogit(
  data = dt, id_col = "id", alt_col = "alt", choice_col = "choice",
  covariate_cols = "x1", random_var_cols = "w1", S = 50L
)
diversion_ratios(fit, "x1")
diversion_ratios(fit, "w1", is_random_coef = TRUE)

```

 elasticities

Compute aggregate elasticities

Description

Computes a $J \times J$ matrix of aggregate elasticities. Entry (i, j) is the percentage change in the probability of choosing alternative i when the attribute of alternative j changes by 1%

Usage

```
elasticities(object, ...)
```

Arguments

`object` A fitted model object.
`...` Additional arguments passed to methods.

Value

A $J \times J$ elasticity matrix with alternative labels.

Examples

```

library(data.table)
set.seed(42)
N <- 50; J <- 3
dt <- data.table(id = rep(1:N, each = J), alt = rep(1:J, N))
dt[, `:=`(x1 = rnorm(.N), x2 = rnorm(.N))]
dt[, choice := 0L]
dt[, choice := sample(c(1L, rep(0L, J - 1))), by = id]
fit <- run_mnlogit(dt, "id", "alt", "choice", c("x1", "x2"))
elasticities(fit, "x1")

```

```
elasticities.choicer_mnl
```

Elasticities for multinomial logit model

Description

Elasticities for multinomial logit model

Usage

```
## S3 method for class 'choicer_mnl'
elasticities(object, elast_var, ...)
```

Arguments

object	A choicer_mnl object fitted with keep_data = TRUE.
elast_var	Variable for elasticity computation: a column name (character) or 1-based index into the design matrix X.
...	Additional arguments (ignored).

Value

A J x J elasticity matrix with alternative labels.

Examples

```
library(data.table)
set.seed(42)
N <- 50; J <- 3
dt <- data.table(id = rep(1:N, each = J), alt = rep(1:J, N))
dt[, `:=`(x1 = rnorm(.N), x2 = rnorm(.N))]
dt[, choice := 0L]
dt[, choice := sample(c(1L, rep(0L, J - 1))), by = id]
fit <- run_mnlogit(dt, "id", "alt", "choice", c("x1", "x2"))
elasticities(fit, "x1")
```

```
elasticities.choicer_mxl
```

Elasticities for mixed logit model

Description

Elasticities for mixed logit model

Usage

```
## S3 method for class 'choicer_mx1'
elasticities(object, elast_var, is_random_coef = FALSE, ...)
```

Arguments

object	A choicer_mx1 object fitted with keep_data = TRUE.
elast_var	Variable for elasticity computation: a column name (character) or 1-based index. Indexes into X columns for fixed coefficients, or W columns for random coefficients (when is_random_coef = TRUE).
is_random_coef	Logical. TRUE if the variable has a random coefficient (is in W), FALSE if fixed (in X). Default FALSE.
...	Additional arguments (ignored).

Value

A J x J elasticity matrix with alternative labels.

Examples

```
library(data.table)
set.seed(42)
N <- 50; J <- 3
dt <- data.table(id = rep(1:N, each = J), alt = rep(1:J, N))
dt[, `:=`(x1 = rnorm(.N), w1 = rnorm(.N))]
dt[, choice := 0L]
dt[, choice := sample(c(1L, rep(0L, J - 1))), by = id]
fit <- run_mxlogit(
  data = dt, id_col = "id", alt_col = "alt", choice_col = "choice",
  covariate_cols = "x1", random_var_cols = "w1", S = 50L
)
elasticities(fit, "x1")
elasticities(fit, "w1", is_random_coef = TRUE)
```

get_halton_normals *Halton draws for mixed logit*

Description

Create halton normal draws in appropriate format for mixed logit estimation

Usage

```
get_halton_normals(S, N, K_w)
```

Arguments

S Number of draws for each choice situation
 N number of choice situations
 K_w dimension of random coefficients (number of columns in W matrix)

Value

K_w x S x N array with halton standard normal draws

Examples

```
draws <- get_halton_normals(S = 50, N = 10, K_w = 2)
dim(draws) # 2 x 50 x 10
```

jacobian_vech_Sigma *Utility to compute analytical Jacobian of random coefficient matrix transformed by vech (dVech(Sigma) / dTheta)*

Description

Utility to compute analytical Jacobian of random coefficient matrix transformed by vech (dVech(Sigma) / dTheta)

Usage

```
jacobian_vech_Sigma(L_params, K_w, rc_correlation = TRUE)
```

Arguments

L_params flattened choleski decomposition version of the random coefficient parameters matrix
 K_w dimension of the random coefficient parameter (symmetric) matrix
 rc_correlation whether random coefficients are correlated

Value

Jacobian (dVech(Sigma) / dTheta)

Examples

```
L_params <- c(log(0.8), 0.2, log(0.6))
J_mat <- jacobian_vech_Sigma(L_params, K_w = 2, rc_correlation = TRUE)
dim(J_mat) # 3 x 3 for K_w=2 correlated
```

logLik.choicer_fit *Extract log-likelihood from a choicer_fit object*

Description

Returns a logLik object, which enables AIC() and BIC() automatically.

Usage

```
## S3 method for class 'choicer_fit'
logLik(object, ...)
```

Arguments

object A choicer_fit object.
 ... Additional arguments (ignored).

Value

A logLik object with df and nobs attributes.

Examples

```
library(data.table)
set.seed(42)
N <- 50; J <- 3
dt <- data.table(id = rep(1:N, each = J), alt = rep(1:J, N))
dt[, `:=`(x1 = rnorm(.N), x2 = rnorm(.N))]
dt[, choice := 0L]
dt[, choice := sample(c(1L, rep(0L, J - 1))), by = id]
fit <- run_mnlogit(dt, "id", "alt", "choice", c("x1", "x2"))
logLik(fit)
AIC(fit)
BIC(fit)
```

mc_asymptotics *Asymptotic diagnostics for a Monte Carlo study*

Description

Consumes a choicer_mc object and returns per-parameter asymptotic diagnostics: Monte Carlo bias (with MC standard error), empirical SD of the estimates, mean of the reported standard errors, SE-to-SD ratio (information-matrix-equality check), Wald coverage at nominal 90 / 95 / 99 percent with Wilson confidence bands, moments of the studentized statistic $z = (\text{theta_hat} - \text{theta}_0) / \text{se}$, and four normality tests on z (Shapiro-Wilk, Anderson-Darling via `gofest::ad.test`, a hand-coded Jarque-Bera statistic, and a one-sample Kolmogorov-Smirnov test against $N(0, 1)$).

Usage

```
mc_asymptotics(
  mc,
  level = 0.95,
  se_col = "se",
  conv_threshold = 0.99,
  se_ratio_threshold_floor = 0.1
)
```

Arguments

mc	A choicer_mc object returned by <code>monte_carlo()</code> .
level	Confidence level for the Wilson bands on coverage rates. Defaults to 0.95.
se_col	Name of the column in <code>mc\$replications</code> to use as the standard-error source. Defaults to "se" (the Hessian-based SE stored by <code>monte_carlo()</code>). Callers that augment replications with an alternative SE flavor (e.g., "se_bhhh" for a BHHH/OPG comparison) can pass that column name to recompute every SE-dependent diagnostic (mean_se, se_ratio, mean_se_w, cov90/95/99, z-moments, normality tests, pass flags) against that flavor. Useful for the information-matrix-equality check in Claim 4 of the MXL validation suite.
conv_threshold	Numeric in $[0, 1]$. Minimum fraction of replications that must converge for the per-parameter <code>pass_convergence</code> flag to be TRUE. The flag compares <code>R_used / R_total</code> (per parameter) against this threshold. Defaults to 0.99.
se_ratio_threshold_floor	Numeric scalar. Minimum half-width for the <code>pass_se_ratio</code> band. The actual band used is $\max(\text{se_ratio_threshold_floor}, 3 * 1.4 / \sqrt{R_used})$, where the $1.4 / \sqrt{R}$ term approximates the large-sample SD of <code>mean_se / sd_emp</code> . The floor guarantees the band is never tighter than the historical hard cutoff. Defaults to 0.10.

Details

Six logical pass / fail flags are attached to every parameter row: `pass_bias` requires $|\text{bias_mc_se}| < 3$; `pass_se_ratio` requires $|\text{se_ratio} - 1|$ to lie within $\max(\text{se_ratio_threshold_floor}, 3 * 1.4 / \sqrt{R_used})$ (a noise-aware band that widens at small `R_used` and tightens to the floor at large `R_used`); `pass_cov95` requires the nominal 95 percent level to lie in the Wilson band for empirical coverage; `pass_skew` requires $|\text{skew_z}| < 0.3$; `pass_kurt` requires excess kurtosis of `z` in $[-0.5, 1.0]$; `pass_convergence` requires the per-parameter convergence rate (`R_used / R_total`) to meet `conv_threshold`.

Non-converged replications are excluded per parameter (reported in `R_excluded`). Winsorized (5 percent / 95 percent) versions of `bias`, `sd_emp`, and `mean_se` are reported in parallel columns (`bias_w`, `sd_emp_w`, `mean_se_w`) so silent outlier exclusion is transparent to the reader. Two robust SE-to-SD ratios accompany the Hessian-mean-based `se_ratio`: `se_ratio_med` (median SE divided by the empirical SD) and `se_ratio_w` (winsorized mean SE divided by the winsorized empirical SD); both stay near 1 when 1-2 replications produce near-singular Hessians that inflate `mean_se`. The companion `se_med` column reports the median per-replication SE used by `se_ratio_med`. Neither robust ratio drives a `pass_*` flag — they are purely informational.

Winsorized z-moment counterparts (`mean_z_w`, `sd_z_w`, `skew_z_w`, `kurt_excess_z_w`) are reported alongside the raw z-moments and feed an additional `pass_z_w` flag (Winsorized skew within the same band as `pass_skew` AND Winsorized excess kurtosis within the same band as `pass_kurt`). A companion `pass_cov95_w` flag is TRUE when either `pass_cov95` is TRUE OR the per-rep Winsorized z-CI (the empirical 2.5 / 97.5 percentiles of the Winsorized z) covers truth-zero. These two flags are designed for boundary scenarios (e.g., near-zero variance components) where a small number of reps with vanishing SE inflate the raw z-moments without indicating an estimator defect.

Value

An object of class `choicer_mc_asymptotics` — a `data.table` with one row per unique parameter and columns documented above — with `meta` attached as an attribute (`attr(x, "meta")`).

Examples

```
sim_fun <- function(seed) simulate_mnl_data(N = 1000, J = 3, seed = seed)
fit_fun <- function(sim) run_mnlogit(
  data = sim$data, id_col = "id", alt_col = "alt", choice_col = "choice",
  covariate_cols = c("x1", "x2"), outside_opt_label = 0L,
  include_outside_option = FALSE, use_asc = TRUE,
  control = list(print_level = 0L)
)
mc <- monte_carlo(sim_fun, fit_fun, R = 50L, seed = 1L, progress = FALSE)
mc_asymptotics(mc)
```

`mnl_diversion_ratios_parallel`

Compute MNL diversion ratios (parallelized over individuals)

Description

Computes the diversion ratio matrix $DR(j \rightarrow k)$, which measures the fraction of demand lost by alternative j that is captured by alternative k . For MNL: $DR(j \rightarrow k) = \frac{\sum_n (w_n * P_{nj} * P_{nk})}{\sum_n (w_n * P_{nj} * (1 - P_{nj}))}$

Usage

```
mnl_diversion_ratios_parallel(
  theta,
  X,
  alt_idx,
  M,
  weights,
  use_asc = TRUE,
  include_outside_option = FALSE
)
```

Arguments

theta	K + J - 1 or K + J vector with model parameters
X	sum(M) x K design matrix with covariates.
alt_idx	sum(M) x 1 vector with indices of alternatives; 1-based indexing
M	N x 1 vector with number of alternatives for each individual
weights	N x 1 vector with weights for each observation
use_asc	whether to use alternative-specific constants
include_outside_option	whether to include outside option

Value

J x J matrix where entry (k, j) = DR(j->k). Diagonal is 0.

Examples

```
library(data.table)
set.seed(42)
N <- 50; J <- 3
dt <- data.table(id = rep(1:N, each = J), alt = rep(1:J, N))
dt[, `:=`(x1 = rnorm(.N), x2 = rnorm(.N))]
dt[, choice := 0L]
dt[, choice := sample(c(1L, rep(0L, J - 1))), by = id]
fit <- run_mnlogit(dt, "id", "alt", "choice", c("x1", "x2"))
dr <- mnl_diversion_ratios_parallel(coef(fit), fit$data$X, fit$data$alt_idx,
  fit$data$M, fit$data$weights)
dr
```

mnl_elasticities_parallel

Compute aggregate elasticities for MNL model

Description

Computes the aggregate elasticity matrix (weighted average of individual elasticities) for the Multinomial Logit model.

Usage

```
mnl_elasticities_parallel(
  theta,
  X,
  alt_idx,
  choice_idx,
  M,
```

```

    weights,
    elast_var_idx,
    use_asc = TRUE,
    include_outside_option = FALSE
  )

```

Arguments

theta	K + J - 1 or K + J vector with model parameters
X	sum(M) x K design matrix with covariates.
alt_idx	sum(M) x 1 vector with indices of alternatives; 1-based indexing
choice_idx	N x 1 vector (kept for API consistency, but not used)
M	N x 1 vector with number of alternatives for each individual
weights	N x 1 vector with weights for each observation
elast_var_idx	1-based index of the column in X for which to compute the elasticity
use_asc	whether to use alternative-specific constants
include_outside_option	whether to include outside option

Value

J x J matrix of aggregate elasticities

Examples

```

library(data.table)
set.seed(42)
N <- 50; J <- 3
dt <- data.table(id = rep(1:N, each = J), alt = rep(1:J, N))
dt[, `:=`(x1 = rnorm(.N), x2 = rnorm(.N))]
dt[, choice := 0L]
dt[, choice := sample(c(1L, rep(0L, J - 1))), by = id]
fit <- run_mnlogit(dt, "id", "alt", "choice", c("x1", "x2"))
elas <- mnl_elasticities_parallel(coef(fit), fit$data$X, fit$data$alt_idx,
  fit$data$choice_idx, fit$data$M, fit$data$weights, elast_var_idx = 1L)
elas

```

mnl_loglik_gradient_parallel

Log-likelihood and gradient for multinomial logit model

Description

Computes the log-likelihood and its gradient for the Multinomial Logit model using OpenMP for parallelization. Allows for inclusion of alternative-specific constants, outside option, and observation weights.

Usage

```
mnl_loglik_gradient_parallel(
  theta,
  X,
  alt_idx,
  choice_idx,
  M,
  weights,
  use_asc = TRUE,
  include_outside_option = FALSE
)
```

Arguments

theta	K + J - 1 or K + J vector with model parameters
X	sum(M) x K design matrix with covariates. Stacks M[i] x K matrices for individual i.
alt_idx	sum(M) x 1 vector with indices of alternatives within each choice set; 1-based indexing
choice_idx	N x 1 vector with indices of chosen alternatives; 1-based indexing relative to X; 0 is used if include_outside_option=True
M	N x 1 vector with number of alternatives for each individual
weights	N x 1 vector with weights for each observation
use_asc	whether to use alternative-specific constants
include_outside_option	whether to include outside option normalized to 0 (if so, the outside option is not included in the data)

Value

List with loglikelihood and gradient evaluated at input arguments

Examples

```
library(data.table)
set.seed(42)
N <- 50; J <- 3
dt <- data.table(id = rep(1:N, each = J), alt = rep(1:J, N))
dt[, `:=`(x1 = rnorm(.N), x2 = rnorm(.N))]
dt[, choice := 0L]
dt[, choice := sample(c(1L, rep(0L, J - 1))), by = id]
d <- prepare_mnl_data(dt, "id", "alt", "choice", c("x1", "x2"))
theta <- rep(0, ncol(d$X) + nrow(d$alt_mapping) - 1)
result <- mnl_loglik_gradient_parallel(theta, d$X, d$alt_idx,
  d$choice_idx, d$M, d$weights)
result$objective # negative log-likelihood
```

mnl_loglik_hessian_parallel

Hessian matrix for multinomial logit model

Description

Hessian matrix for multinomial logit model

Usage

```
mnl_loglik_hessian_parallel(
  theta,
  X,
  alt_idx,
  choice_idx,
  M,
  weights,
  use_asc = TRUE,
  include_outside_option = FALSE
)
```

Arguments

theta	K + J - 1 or K + J vector with model parameters
X	sum(M) x K design matrix with covariates. Stacks M[i] x K matrices for individual i.
alt_idx	sum(M) x 1 vector with indices of alternatives within each choice set; 1-based indexing
choice_idx	N x 1 vector with indices of chosen alternatives; 1-based indexing relative to X; 0 is used if include_outside_option=True
M	N x 1 vector with number of alternatives for each individual
weights	N x 1 vector with weights for each observation
use_asc	whether to use alternative-specific constants
include_outside_option	whether to include outside option normalized to 0 (if so, the outside option is not included in the data)

Value

Hessian matrix of the negative log-likelihood

Examples

```

library(data.table)
set.seed(42)
N <- 50; J <- 3
dt <- data.table(id = rep(1:N, each = J), alt = rep(1:J, N))
dt[, `:=`(x1 = rnorm(.N), x2 = rnorm(.N))]
dt[, choice := 0L]
dt[, choice := sample(c(1L, rep(0L, J - 1))), by = id]
fit <- run_mnlogit(dt, "id", "alt", "choice", c("x1", "x2"))
H <- mnl_loglik_hessian_parallel(coef(fit), fit$data$X, fit$data$salt_idx,
  fit$data$choice_idx, fit$data$M, fit$data$weights)
dim(H)

```

mnl_predict

Prediction of choice probabilities and utilities based on fitted model

Description

Prediction of choice probabilities and utilities based on fitted model

Usage

```

mnl_predict(
  theta,
  X,
  alt_idx,
  M,
  use_asc = TRUE,
  include_outside_option = FALSE
)

```

Arguments

theta	K + J - 1 or K + J vector with model parameters
X	sum(M) x K design matrix with covariates. Stacks M[i] x K matrices for individual i.
alt_idx	sum(M) x 1 vector with indices of alternatives within each choice set; 1-based indexing
M	N x 1 vector with number of alternatives for each individual
use_asc	whether to use alternative-specific constants
include_outside_option	whether to include outside option normalized to 0 (if so, the outside option is not included in the data)

Value

List with choice probability and utility for each choice situation evaluated at input arguments

Examples

```
library(data.table)
set.seed(42)
N <- 50; J <- 3
dt <- data.table(id = rep(1:N, each = J), alt = rep(1:J, N))
dt[, `:=`(x1 = rnorm(.N), x2 = rnorm(.N))]
dt[, choice := 0L]
dt[, choice := sample(c(1L, rep(0L, J - 1))), by = id]
fit <- run_mnlogit(dt, "id", "alt", "choice", c("x1", "x2"))
pred <- mnl_predict(coef(fit), fit$data$X, fit$data$alt_idx,
  fit$data$M, use_asc = TRUE)
head(pred$choice_prob)
```

mnl_predict_shares *Prediction of market shares based on fitted model*

Description

Prediction of market shares based on fitted model

Usage

```
mnl_predict_shares(
  theta,
  X,
  alt_idx,
  M,
  weights,
  use_asc = TRUE,
  include_outside_option = FALSE
)
```

Arguments

theta	K + J - 1 or K + J vector with model parameters
X	sum(M) x K design matrix with covariates. Stacks M[i] x K matrices for individual i.
alt_idx	sum(M) x 1 vector with indices of alternatives within each choice set; 1-based indexing
M	N x 1 vector with number of alternatives for each individual
weights	N x 1 vector with weights for each observation

use_asc whether to use alternative-specific constants
include_outside_option
 whether to include outside option normalized to 0 (if so, the outside option is not included in the data)

Value

vector with predicted market shares for each alternative

Examples

```

library(data.table)
set.seed(42)
N <- 50; J <- 3
dt <- data.table(id = rep(1:N, each = J), alt = rep(1:J, N))
dt[, `:=`(x1 = rnorm(.N), x2 = rnorm(.N))]
dt[, choice := 0L]
dt[, choice := sample(c(1L, rep(0L, J - 1))), by = id]
fit <- run_mnlogit(dt, "id", "alt", "choice", c("x1", "x2"))
shares <- mnl_predict_shares(coef(fit), fit$data[X, fit$data$alt_idx,
  fit$data$M, fit$data$weights, use_asc = TRUE])
shares

```

monte_carlo

Monte Carlo parameter recovery

Description

Replicates a (DGP → fit) cycle R times with independent seeds and collects per-parameter estimates, standard errors, bias, and coverage. Returns a `choicer_mc` object; call `summary()` for aggregated statistics (mean estimate, bias, RMSE, coverage rate, convergence rate).

Usage

```

monte_carlo(
  sim_fun,
  fit_fun,
  R = 100,
  seed = 1L,
  parallel = FALSE,
  progress = TRUE,
  ...
)

```

Arguments

sim_fun	Function of seed returning a choicer_sim.
fit_fun	Function of a choicer_sim returning a choicer_fit.
R	Number of replications.
seed	Base integer seed. Replication r uses seed + r - 1L.
parallel	Logical; if TRUE and future.apply is available, run replications in parallel using the user's active future::plan().
progress	Logical; print a one-line progress update per iteration in serial mode. Ignored when parallel = TRUE.
...	Unused.

Details

Each iteration calls `sim_fun(seed = seed + r - 1L)`, then `fit_fun(sim)`. Write `sim_fun` as a closure that captures `N`, `J`, and other DGP settings and forwards `seed`. Write `fit_fun` as a closure that takes a `choicer_sim` and returns a fitted `choicer_fit` object, wrapping any data-preparation, draws, or optimizer-control setup.

Value

A `choicer_mc` object: a list with elements `replications` (a long `data.table` with one row per estimated parameter per replication) and `meta` (run metadata).

Examples

```
sim_fun <- function(seed) simulate_mnl_data(N = 1000, J = 4, seed = seed)
fit_fun <- function(sim) run_mnlogit(
  data = sim$data, id_col = "id", alt_col = "alt", choice_col = "choice",
  covariate_cols = c("x1", "x2"), outside_opt_label = 0L,
  include_outside_option = FALSE, use_asc = TRUE,
  control = list(print_level = 0L)
)
mc <- monte_carlo(sim_fun, fit_fun, R = 5, seed = 1L, progress = FALSE)
summary(mc)
```

mxl_bhhh_parallel	<i>BHHH (outer product of gradients) information matrix for Mixed Logit</i>
-------------------	---

Description

Computes the BHHH approximation to the observed information matrix for the Mixed Logit model: $H_{BHHH} = \sum_i w_i \cdot s_i s_i^\top$, where s_i is the per-individual score (gradient of $\log \bar{P}_i$). This outer product of gradients (OPG) estimator provides an alternative to the analytical Hessian for standard error computation that scales to large problems where the analytical Hessian is infeasible (e.g., many alternatives or simulation draws).

Usage

```
mx1_bhhh_parallel(
  theta,
  X,
  W,
  alt_idx,
  choice_idx,
  M,
  weights,
  eta_draws,
  rc_dist,
  rc_correlation = TRUE,
  rc_mean = FALSE,
  use_asc = TRUE,
  include_outside_option = FALSE
)
```

Arguments

theta	vector collecting model parameters (beta, mu, L, delta (ASCs))
X	design matrix for covariates with fixed coefficients; $\text{sum}(M_i) \times K_x$
W	design matrix for covariates with random coefficients; $\text{sum}(M_i) \times K_w$ or $J \times K_w$
alt_idx	$\text{sum}(M) \times 1$ vector with indices of alternatives within each choice set; 1-based indexing
choice_idx	$N \times 1$ vector with indices of chosen alternatives; 1-based indexing relative to X; 0 is used if include_outside_option=True
M	$N \times 1$ vector with number of alternatives for each individual
weights	$N \times 1$ vector with weights for each observation
eta_draws	Array with choice situation draws; $K_w \times S \times N$
rc_dist	$K_w \times 1$ integer vector indicating distribution of random coefficients: 0 = normal, 1 = log-normal
rc_correlation	whether random coefficients should be correlated
rc_mean	whether to estimate means for random coefficients.
use_asc	whether to use alternative-specific constants.
include_outside_option	whether to include outside option normalized to 0 (if so, the outside option is not included in the data)

Value

$n_params \times n_params$ PSD matrix representing the observed information matrix estimated by the outer product of gradients (same sign convention as the negated Hessian returned by `mx1_hessian_parallel`, so it can be inverted directly to obtain vcov).

Note

The BHHH/OPG estimator is only asymptotically equivalent to the Hessian-based information matrix at the true MLE. In finite samples it can underestimate standard errors, particularly when the model is mis-specified or away from the optimum.

Examples

```
library(data.table)
set.seed(42)
N <- 50; J <- 3
dt <- data.table(id = rep(1:N, each = J), alt = rep(1:J, N))
dt[, `:=`(x1 = rnorm(.N), w1 = rnorm(.N))]
dt[, choice := 0L]
dt[, choice := sample(c(1L, rep(0L, J - 1))), by = id]
d <- prepare_mxl_data(dt, "id", "alt", "choice", "x1", "w1")
eta <- get_halton_normals(50, d$N, ncol(d$W))
theta <- rep(0, ncol(d$X) + ncol(d$W) + nrow(d$alt_mapping) - 1)
H <- mxl_bhhh_parallel(theta, d$X, d$W, d$alt_idx, d$choice_idx,
  d$M, d$weights, eta, rc_dist = rep(0L, ncol(d$W)),
  rc_correlation = FALSE, rc_mean = FALSE)
dim(H)
```

mxl_blp_contraction *BLP contraction mapping for mixed logit*

Description

Finds the ASC (delta) parameters such that predicted market shares match target shares, using the contraction mapping of Berry, Levinsohn, and Pakes (1995).

Usage

```
mxl_blp_contraction(
  delta,
  target_shares,
  X,
  W,
  beta,
  mu,
  L_params,
  alt_idx,
  M,
  weights,
  eta_draws,
  rc_dist,
  rc_correlation = TRUE,
  rc_mean = FALSE,
```

```

include_outside_option = FALSE,
tol = 1e-08,
max_iter = 1000L
)

```

Arguments

delta	J-1 or J vector with initial guess for deltas (ASCs)
target_shares	J vector with target market shares
X	design matrix for fixed coefficients; $\text{sum}(M_i) \times K_x$
W	design matrix for random coefficients; $\text{sum}(M_i) \times K_w$ or $J \times K_w$
beta	K_x vector with fixed coefficients
mu	K_w vector with mean parameters (raw, will be transformed if log-normal)
L_params	Cholesky parameters vector
alt_idx	$\text{sum}(M) \times 1$ vector with indices of alternatives; 1-based indexing
M	$N \times 1$ vector with number of alternatives for each individual
weights	$N \times 1$ vector with weights for each observation
eta_draws	Array with draws; $K_w \times S \times N$
rc_dist	K_w vector indicating distribution (0=normal, 1=log-normal)
rc_correlation	whether random coefficients are correlated
rc_mean	whether mu parameters represent means (TRUE) or are zero (FALSE)
include_outside_option	whether outside option is included
tol	convergence tolerance (default 1e-8)
max_iter	maximum iterations (default 1000)

Value

vector with converged delta (ASC) values

Examples

```

library(data.table)
set.seed(42)
N <- 50; J <- 3
dt <- data.table(id = rep(1:N, each = J), alt = rep(1:J, N))
dt[, `:=`(x1 = rnorm(.N), w1 = rnorm(.N))]
dt[, choice := 0L]
dt[, choice := sample(c(1L, rep(0L, J - 1))), by = id]
d <- prepare_mxl_data(dt, "id", "alt", "choice", "x1", "w1")
eta <- get_halton_normals(50, d$N, ncol(d$W))
fit <- run_mxlogit(input_data = d, eta_draws = eta)
pm <- fit$param_map
delta <- mxl_blp_contraction(rep(0, J), rep(1/J, J), d$X, d$W,
  coef(fit)[pm$beta], rep(0, ncol(d$W)), coef(fit)[pm$sigma],

```

```
d$alt_idx, d$M, d$weights, eta, rc_dist = rep(0L, ncol(d$W)),
rc_correlation = FALSE, rc_mean = FALSE)
delta
```

```
mxl_diversion_ratios_parallel
```

Diversion ratios for Mixed Logit (simulated, derivative-based)

Description

Computes the matrix of attribute-based diversion ratios for a fitted Mixed Logit model. $DR(k, j)$ is the fraction of demand lost by alternative j that is captured by alternative k when a marginal change in alternative j 's `elast_var` attribute reduces s_j .

Usage

```
mxl_diversion_ratios_parallel(
  theta,
  X,
  W,
  alt_idx,
  M,
  weights,
  eta_draws,
  rc_dist,
  elast_var_idx,
  is_random_coef,
  rc_correlation = TRUE,
  rc_mean = FALSE,
  use_asc = TRUE,
  include_outside_option = FALSE
)
```

Arguments

<code>theta</code>	parameter vector (beta, [mu], L, delta)
<code>X</code>	design matrix for fixed coefficients; $\text{sum}(M_i) \times K_x$
<code>W</code>	design matrix for random coefficients; $\text{sum}(M_i) \times K_w$ or $J \times K_w$
<code>alt_idx</code>	$\text{sum}(M) \times 1$ vector with indices of alternatives; 1-based indexing
<code>M</code>	$N \times 1$ vector with number of alternatives for each individual
<code>weights</code>	$N \times 1$ vector with weights for each observation
<code>eta_draws</code>	Array with draws; $K_w \times S \times N$
<code>rc_dist</code>	K_w vector indicating distribution (0=normal, 1=log-normal)
<code>elast_var_idx</code>	1-based index of the perturbed variable

is_random_coef TRUE if the variable is in W (random coef), FALSE if in X (fixed)
 rc_correlation whether random coefficients are correlated
 rc_mean whether mu parameters are estimated
 use_asc whether ASCs are included
 include_outside_option whether outside option is included

Details

In MNL the per-draw realized coefficient is a constant, so it cancels in the ratio and the result is independent of the variable chosen. In MXL, the realized coefficient β_{ik}^s varies across individuals and draws, so the diversion ratio depends on which attribute is perturbed. For a variable with a fixed coefficient the dependence again vanishes (the constant cancels); for a random-coefficient variable it does not.

Value

J x J (or (J+1) x (J+1)) matrix of diversion ratios with zero diagonal.

mxl_elasticities_parallel

Compute aggregate elasticities for mixed logit model

Description

Computes the aggregate elasticity matrix (weighted average of individual elasticities) for the Mixed Logit model. The elasticity E(i,j) represents the percentage change in the probability of choosing alternative i when the attribute of alternative j changes by 1%.

Usage

```

mxl_elasticities_parallel(
  theta,
  X,
  W,
  alt_idx,
  choice_idx,
  M,
  weights,
  eta_draws,
  rc_dist,
  elast_var_idx,
  is_random_coef,
  rc_correlation = TRUE,
  rc_mean = FALSE,
  use_asc = TRUE,
  include_outside_option = FALSE
)

```

Arguments

theta	parameter vector (beta, [mu], L, delta)
X	design matrix for fixed coefficients; sum(M_i) x K_x
W	design matrix for random coefficients; sum(M_i) x K_w or J x K_w
alt_idx	sum(M) x 1 vector with indices of alternatives; 1-based indexing
choice_idx	N x 1 vector (kept for API consistency, not used)
M	N x 1 vector with number of alternatives for each individual
weights	N x 1 vector with weights for each observation
eta_draws	Array with draws; K_w x S x N
rc_dist	K_w vector indicating distribution (0=normal, 1=log-normal)
elast_var_idx	1-based index of the variable for elasticity computation
is_random_coef	TRUE if variable is in W (random coef), FALSE if in X (fixed coef)
rc_correlation	whether random coefficients are correlated
rc_mean	whether mu parameters are estimated
use_asc	whether ASCs are included
include_outside_option	whether outside option is included

Value

J x J matrix of aggregate elasticities

Examples

```
library(data.table)
set.seed(42)
N <- 50; J <- 3
dt <- data.table(id = rep(1:N, each = J), alt = rep(1:J, N))
dt[, `:=`(x1 = rnorm(.N), w1 = rnorm(.N))]
dt[, choice := 0L]
dt[, choice := sample(c(1L, rep(0L, J - 1))), by = id]
d <- prepare_mxldata(dt, "id", "alt", "choice", "x1", "w1")
eta <- get_halton_normals(50, d$N, ncol(d$W))
fit <- run_mxlogit(input_data = d, eta_draws = eta)
elas <- mxl_elasticities_parallel(coef(fit), d$X, d$W, d$alt_idx,
  d$choice_idx, d$M, d$weights, eta, rc_dist = rep(0L, ncol(d$W)),
  elast_var_idx = 1L, is_random_coef = FALSE,
  rc_correlation = FALSE, rc_mean = FALSE)
elas
```

 mxl_hessian_parallel *Analytical Hessian of the log-likelihood v2*

Description

Computes the Hessian of the log-likelihood for the Mixed Logit model using OpenMP for parallelization. Mirrors the parameters of mxl_loglik_gradient_parallel.

Usage

```
mxl_hessian_parallel(
  theta,
  X,
  W,
  alt_idx,
  choice_idx,
  M,
  weights,
  eta_draws,
  rc_dist,
  rc_correlation = TRUE,
  rc_mean = FALSE,
  use_asc = TRUE,
  include_outside_option = FALSE
)
```

Arguments

theta	vector collecting model parameters (beta, mu, L, delta (ASCs))
X	design matrix for covariates with fixed coefficients; $\text{sum}(M_i) \times K_x$
W	design matrix for covariates with random coefficients; $\text{sum}(M_i) \times K_w$ or $J \times K_w$
alt_idx	$\text{sum}(M) \times 1$ vector with indices of alternatives within each choice set; 1-based indexing
choice_idx	$N \times 1$ vector with indices of chosen alternatives; 1-based indexing relative to X; 0 is used if include_outside_option=True
M	$N \times 1$ vector with number of alternatives for each individual
weights	$N \times 1$ vector with weights for each observation
eta_draws	Array with choice situation draws; $K_w \times S \times N$
rc_dist	$K_w \times 1$ integer vector indicating distribution of random coefficients: 0 = normal, 1 = log-normal
rc_correlation	whether random coefficients should be correlated
rc_mean	whether to estimate means for random coefficients.
use_asc	whether to use alternative-specific constants.

`include_outside_option`
 whether to include outside option normalized to 0 (if so, the outside option is not included in the data)

Value

Hessian evaluated at input arguments

Note

For log-normal random coefficients (`rc_dist=1`) with `rc_mean=TRUE`, the distribution is a shifted log-normal: $\beta_k = \exp(\mu_k) + \exp(L_k * \eta)$, where $\exp(\mu_k)$ shifts the location and $\exp(L_k * \eta) \sim \text{LogNormal}(0, \sigma_k^2)$. This differs from the textbook parameterization $\exp(\mu_k + L_k * \eta)$.

Examples

```
library(data.table)
set.seed(42)
N <- 50; J <- 3
dt <- data.table(id = rep(1:N, each = J), alt = rep(1:J, N))
dt[, `:=`(x1 = rnorm(.N), w1 = rnorm(.N))]
dt[, choice := 0L]
dt[, choice := sample(c(1L, rep(0L, J - 1))), by = id]
d <- prepare_mxl_data(dt, "id", "alt", "choice", "x1", "w1")
eta <- get_halton_normals(50, d$N, ncol(d$W))
theta <- rep(0, ncol(d$X) + ncol(d$W) + nrow(d$alt_mapping) - 1)
H <- mxl_hessian_parallel(theta, d$X, d$W, d$alt_idx, d$choice_idx,
  d$M, d$weights, eta, rc_dist = rep(0L, ncol(d$W)),
  rc_correlation = FALSE, rc_mean = FALSE)
dim(H)
```

mxl_loglik_gradient_parallel

Log-likelihood and gradient for Mixed Logit

Description

Computes the log-likelihood and its gradient for the Mixed Logit model using OpenMP for parallelization. Allows for inclusion of alternative-specific constants, outside option, observation weights, correlated random coefficients.

Usage

```
mxl_loglik_gradient_parallel(
  theta,
  X,
  W,
```

```

    alt_idx,
    choice_idx,
    M,
    weights,
    eta_draws,
    rc_dist,
    rc_correlation = TRUE,
    rc_mean = FALSE,
    use_asc = TRUE,
    include_outside_option = FALSE
  )

```

Arguments

theta	vector collecting model parameters (beta, mu, L, delta (ASCs))
X	design matrix for covariates with fixed coefficients; $\text{sum}(M_i) \times K_x$
W	design matrix for covariates with random coefficients; $\text{sum}(M_i) \times K_w$ or $J \times K_w$
alt_idx	$\text{sum}(M) \times 1$ vector with indices of alternatives within each choice set; 1-based indexing
choice_idx	$N \times 1$ vector with indices of chosen alternatives; 1-based indexing relative to X; 0 is used if include_outside_option=True
M	$N \times 1$ vector with number of alternatives for each individual
weights	$N \times 1$ vector with weights for each observation
eta_draws	Array with choice situation draws; $K_w \times S \times N$
rc_dist	$K_w \times 1$ integer vector indicating distribution of random coefficients: 0 = normal, 1 = log-normal
rc_correlation	whether random coefficients should be correlated
rc_mean	whether to estimate means for random coefficients. If so, mean parameters (mu) should be included in theta after beta parameters.
use_asc	whether to use alternative-specific constants. If so, parameters should be included in theta after beta and L (and mu, if applicable).
include_outside_option	whether to include outside option normalized to 0 (if so, the outside option is not included in the data)

Value

List with loglikelihood and gradient evaluated at input arguments

Note

For log-normal random coefficients (rc_dist=1) with rc_mean=TRUE, the distribution is a shifted log-normal: $\beta_k = \exp(\mu_k) + \exp(L_k * \eta)$, where $\exp(\mu_k)$ shifts the location and $\exp(L_k * \eta) \sim \text{LogNormal}(0, \sigma_k^2)$. This differs from the textbook parameterization $\exp(\mu_k + L_k * \eta)$.

Examples

```

library(data.table)
set.seed(42)
N <- 50; J <- 3
dt <- data.table(id = rep(1:N, each = J), alt = rep(1:J, N))
dt[, `:=`(x1 = rnorm(.N), w1 = rnorm(.N))]
dt[, choice := 0L]
dt[, choice := sample(c(1L, rep(0L, J - 1))), by = id]
d <- prepare_mxl_data(dt, "id", "alt", "choice", "x1", "w1")
eta <- get_halton_normals(50, d$N, ncol(d$W))
K_x <- ncol(d$X); K_w <- ncol(d$W); J <- nrow(d$alt_mapping)
theta <- rep(0, K_x + K_w + J - 1)
result <- mxl_loglik_gradient_parallel(theta, d$X, d$W, d$alt_idx,
  d$choice_idx, d$M, d$weights, eta, rc_dist = rep(0L, K_w),
  rc_correlation = FALSE, rc_mean = FALSE)
result$objective

```

mxl_predict

*Per-observation simulated choice probabilities for Mixed Logit***Description**

Returns the simulated choice probability for each (individual, alternative) row of X, averaged over the supplied Halton draws. Mirrors `mnl_predict`.

Usage

```

mxl_predict(
  theta,
  X,
  W,
  alt_idx,
  M,
  eta_draws,
  rc_dist,
  rc_correlation = TRUE,
  rc_mean = FALSE,
  use_asc = TRUE,
  include_outside_option = FALSE
)

```

Arguments

theta	parameter vector (beta, [mu], L, delta)
X	design matrix for fixed coefficients; $\text{sum}(M_i) \times K_x$
W	design matrix for random coefficients; $\text{sum}(M_i) \times K_w$ or $J \times K_w$

alt_idx	sum(M) x 1 vector with indices of alternatives; 1-based indexing
M	N x 1 vector with number of alternatives for each individual
eta_draws	Array with draws; K_w x S x N
rc_dist	K_w vector indicating distribution (0=normal, 1=log-normal)
rc_correlation	whether random coefficients are correlated
rc_mean	whether mu parameters are estimated
use_asc	whether ASCs are included
include_outside_option	whether the outside option is present

Value

List with choice_prob (length sum(M)), utility (length sum(M), simulated mean of the deterministic + W*gamma component), and, when include_outside_option = TRUE, choice_prob_outside (length N).

mxl_predict_shares	<i>Predicted aggregate market shares for Mixed Logit</i>
--------------------	--

Description

Exported wrapper around the internal mxl_predict_shares_internal. Parses theta using the standard parameter ordering and returns the simulated weighted-average market shares.

Usage

```
mxl_predict_shares(
  theta,
  X,
  W,
  alt_idx,
  M,
  weights,
  eta_draws,
  rc_dist,
  rc_correlation = TRUE,
  rc_mean = FALSE,
  use_asc = TRUE,
  include_outside_option = FALSE
)
```

Arguments

theta	parameter vector (beta, [mu], L, delta)
X	design matrix for fixed coefficients; $\text{sum}(M_i) \times K_x$
W	design matrix for random coefficients; $\text{sum}(M_i) \times K_w$ or $J \times K_w$
alt_idx	$\text{sum}(M)$ x 1 vector with indices of alternatives; 1-based indexing
M	$N \times 1$ vector with number of alternatives for each individual
weights	$N \times 1$ vector with weights for each observation
eta_draws	Array with draws; $K_w \times S \times N$
rc_dist	K_w vector indicating distribution (0=normal, 1=log-normal)
rc_correlation	whether random coefficients are correlated
rc_mean	whether mu parameters are estimated
use_asc	whether ASCs are included
include_outside_option	whether outside option is included

Value

Vector of length J (or $J+1$ with outside option) of predicted shares.

new_choicer_sim	<i>Construct a choicer_sim object</i>
-----------------	---------------------------------------

Description

Wraps simulated data, true parameter values, and DGP settings into a classed list. Returned by [simulate_mnl_data\(\)](#), [simulate_mx1_data\(\)](#), and [simulate_nl_data\(\)](#), and consumed by [recovery_table\(\)](#).

Usage

```
new_choicer_sim(data, true_params, settings, model)
```

Arguments

data	A data.table of simulated choice observations.
true_params	Named list of true DGP parameters (e.g. beta, delta, Sigma, mu, lambdas).
settings	Named list of DGP settings (e.g. N, J, K_x).
model	Character scalar: "mnl", "mx1", or "nl".

Value

A list of class choicer_sim.

 nl_loglik_gradient_parallel

Log-likelihood and gradient for Nested Logit model

Description

Computes the log-likelihood and its gradient for the Nested Logit model using OpenMP for parallelization. Especially handles singleton nests by fixing their lambda parameters to 1. Only non-singleton nests have a inclusive value coefficient estimated in theta.

Usage

```
nl_loglik_gradient_parallel(
  theta,
  X,
  alt_idx,
  choice_idx,
  nest_idx,
  M,
  weights,
  use_asc = TRUE,
  include_outside_option = FALSE
)
```

Arguments

theta	(K + n_non_singleton_nests + n_delta) vector with model parameters. Order: [beta (K), lambda (n_non_singleton_nests), delta (n_delta)]
X	sum(M) x K design matrix with covariates.
alt_idx	sum(M) x 1 vector with indices of alternatives; 1-based indexing.
choice_idx	N x 1 vector with indices of chosen alternatives; 0 for outside option, 1-based index relative to rows in X _i otherwise.
nest_idx	J x 1 vector with indices of nests for each alternative; 1-based indexing (1 to n_nests).
M	N x 1 vector with number of alternatives for each individual.
weights	N x 1 vector with weights for each observation.
use_asc	whether to use alternative-specific constants.
include_outside_option	whether to include outside option normalized to V=0, lambda=1.

Value

List with loglikelihood and gradient evaluated at input arguments

Examples

```

library(data.table)
set.seed(42)
N <- 50; J <- 4
dt <- data.table(id = rep(1:N, each = J), alt = rep(1:J, N))
dt[, `:=`(x1 = rnorm(.N), x2 = rnorm(.N))]
dt[, nest := ifelse(alt <= 2, "A", "B")]
dt[, choice := 0L]
dt[, choice := sample(c(1L, rep(0L, J - 1))), by = id]
d <- prepare_nl_data(dt, "id", "alt", "choice", c("x1", "x2"), "nest")
K_x <- ncol(d$X); K_l <- length(unique(d$nest_idx))
theta <- c(rep(0, K_x), rep(0.5, K_l), rep(0, J - 1))
result <- nl_loglik_gradient_parallel(theta, d$X, d$alt_idx,
  d$choice_idx, d$nest_idx, d$M, d$weights)
result$objective

```

nl_loglik_numeric_hessian

Numerical Hessian of the log-likelihood via finite differences

Description

Numerical Hessian of the log-likelihood via finite differences

Usage

```

nl_loglik_numeric_hessian(
  theta,
  X,
  alt_idx,
  choice_idx,
  nest_idx,
  M,
  weights,
  use_asc = TRUE,
  include_outside_option = FALSE,
  eps = 1e-06
)

```

Arguments

theta	(K + n_delta + n_nests) vector with model parameters. Order: [beta (K), delta (n_delta), lambda (n_lambda)]
X	sum(M) x K design matrix with covariates.
alt_idx	sum(M) x 1 vector with indices of alternatives; 1-based indexing.
choice_idx	N x 1 vector with indices of chosen alternatives; 0 for outside option, 1-based index relative to rows in X_i otherwise.

nest_idx	J x 1 vector with indices of nests for each alternative; 1-based indexing (1 to n_nests).
M	N x 1 vector with number of alternatives for each individual.
weights	N x 1 vector with weights for each observation.
use_asc	whether to use alternative-specific constants.
include_outside_option	whether to include outside option normalized to V=0, lambda=1.
eps	finite difference step size

Value

Hessian evaluated at input arguments

Examples

```
library(data.table)
set.seed(42)
N <- 50; J <- 4
dt <- data.table(id = rep(1:N, each = J), alt = rep(1:J, N))
dt[, `:=`(x1 = rnorm(.N), x2 = rnorm(.N))]
dt[, nest := ifelse(alt <= 2, "A", "B")]
dt[, choice := 0L]
dt[, choice := sample(c(1L, rep(0L, J - 1))), by = id]
d <- prepare_nl_data(dt, "id", "alt", "choice", c("x1", "x2"), "nest")
K_x <- ncol(d[X]); K_1 <- length(unique(d$nest_idx))
theta <- c(rep(0, K_x), rep(0.5, K_1), rep(0, J - 1))
H <- nl_loglik_numeric_hessian(theta, d[X], d$alt_idx, d$choice_idx,
  d$nest_idx, d$M, d$weights)
dim(H)
```

nobs.chooser_fit	<i>Extract number of observations from a chooser_fit object</i>
------------------	---

Description

Extract number of observations from a chooser_fit object

Usage

```
## S3 method for class 'chooser_fit'
nobs(object, ...)
```

Arguments

object	A chooser_fit object.
...	Additional arguments (ignored).

Value

Integer number of choice situations.

Examples

```
library(data.table)
set.seed(42)
N <- 50; J <- 3
dt <- data.table(id = rep(1:N, each = J), alt = rep(1:J, N))
dt[, `:=`(x1 = rnorm(.N), x2 = rnorm(.N))]
dt[, choice := 0L]
dt[, choice := sample(c(1L, rep(0L, J - 1))), by = id]
fit <- run_mnlogit(dt, "id", "alt", "choice", c("x1", "x2"))
nobs(fit)
```

predict.choicer_mnl *Predict from a multinomial logit model*

Description

Computes choice probabilities or aggregate market shares.

Usage

```
## S3 method for class 'choicer_mnl'
predict(object, type = c("probabilities", "shares"), ...)
```

Arguments

object	A choicer_mnl object.
type	One of "probabilities" (individual-level choice probabilities) or "shares" (aggregate market shares).
...	Additional arguments (ignored).

Value

For "probabilities": a list with choice_prob and utility vectors. For "shares": a named numeric vector of market shares per alternative.

Examples

```
library(data.table)
set.seed(42)
N <- 50; J <- 3
dt <- data.table(id = rep(1:N, each = J), alt = rep(1:J, N))
dt[, `:=`(x1 = rnorm(.N), x2 = rnorm(.N))]
dt[, choice := 0L]
```

```
dt[, choice := sample(c(1L, rep(0L, J - 1))), by = id]
fit <- run_mnlogit(dt, "id", "alt", "choice", c("x1", "x2"))
predict(fit, type = "shares")
predict(fit, type = "probabilities")
```

predict.choicer_mxl *Predict from a mixed logit model*

Description

Computes simulated choice probabilities or aggregate market shares using stored Halton draws.

Usage

```
## S3 method for class 'choicer_mxl'
predict(object, type = c("probabilities", "shares"), ...)
```

Arguments

object	A choicer_mxl object.
type	Either "probabilities" (per-observation simulated choice probabilities) or "shares" (aggregate simulated market shares).
...	Additional arguments (ignored).

Value

For "probabilities": a list with choice_prob and utility vectors averaged across simulation draws.
 For "shares": a named numeric vector of simulated market shares per alternative.

Examples

```
library(data.table)
set.seed(42)
N <- 50; J <- 3
dt <- data.table(id = rep(1:N, each = J), alt = rep(1:J, N))
dt[, `:=`(x1 = rnorm(.N), w1 = rnorm(.N))]
dt[, choice := 0L]
dt[, choice := sample(c(1L, rep(0L, J - 1))), by = id]
fit <- run_mxlogit(
  data = dt, id_col = "id", alt_col = "alt", choice_col = "choice",
  covariate_cols = "x1", random_var_cols = "w1", S = 50L
)
predict(fit, type = "shares")
predict(fit, type = "probabilities")
```

```
prepare_mnl_data      Prepare inputs for mnl_loglik_gradient_parallel()
```

Description

Prepares and validates inputs for multinomial logit estimation routine.

Usage

```
prepare_mnl_data(
  data,
  id_col,
  alt_col,
  choice_col,
  covariate_cols,
  weights = NULL,
  outside_opt_label = NULL,
  include_outside_option = FALSE
)
```

Arguments

<code>data</code>	Data frame containing choice data.
<code>id_col</code>	Name of the column identifying choice situations (individuals).
<code>alt_col</code>	Name of the column identifying alternatives.
<code>choice_col</code>	Name of the column indicating chosen alternative (1 = chosen, 0 = not chosen).
<code>covariate_cols</code>	Vector of names of columns to be used as covariates.
<code>weights</code>	Optional vector of weights for each choice situation. If NULL, equal weights are used.
<code>outside_opt_label</code>	Label for the outside option (if any). If NULL, no outside option is assumed.
<code>include_outside_option</code>	Logical indicating whether to include an outside option in the model.

Value

A list containing:

- `X`: Design matrix (sum(M) x K).
- `alt_idx`: Integer vector of alternative indices.
- `choice_idx`: Integer vector of chosen alternative indices.
- `M`: Integer vector with number of alternatives per choice situation.
- `N`: Number of choice situations.
- `weights`: Vector of weights.

- `include_outside_option`: Logical flag.
- `alt_mapping`: Data.table mapping alternatives to summary statistics.
- `dropped_cols`: Names of columns dropped due to collinearity, if any.

Examples

```
library(data.table)
set.seed(42)
N <- 50; J <- 3
dt <- data.table(id = rep(1:N, each = J), alt = rep(1:J, N))
dt[, `:=`(x1 = rnorm(.N), x2 = rnorm(.N))]
dt[, choice := 0L]
dt[, choice := sample(c(1L, rep(0L, J - 1))), by = id]
input <- prepare_mnl_data(dt, "id", "alt", "choice", c("x1", "x2"))
str(input$X)
input$alt_mapping
```

prepare_mxl_data	<i>Prepare inputs for mxl_loglik_gradient_parallel()</i>
------------------	--

Description

Prepares and validates inputs for mixed logit estimation routine.

Usage

```
prepare_mxl_data(
  data,
  id_col,
  alt_col,
  choice_col,
  covariate_cols,
  random_var_cols,
  weights = NULL,
  outside_opt_label = NULL,
  include_outside_option = FALSE,
  rc_correlation = FALSE
)
```

Arguments

<code>data</code>	Data frame containing choice data
<code>id_col</code>	Name of the column identifying choice situations (individuals)
<code>alt_col</code>	Name of the column identifying alternatives
<code>choice_col</code>	Name of the column indicating chosen alternative (1 = chosen, 0 = not chosen)
<code>covariate_cols</code>	Vector of names of columns to be used as covariates

random_var_cols	Vector of names of columns to be used as random variables
weights	Optional vector of weights for each choice situation. If NULL, equal weights are used.
outside_opt_label	Label for the outside option (if any). If NULL, no outside option is assumed.
include_outside_option	Logical indicating whether to include an outside option in the model.
rc_correlation	Logical indicating whether random coefficients are correlated. Default is FALSE.

Value

A choicer_data_mxl object (list) containing:

- X: Fixed-coefficient design matrix (sum(M) x K_x).
- W: Random-coefficient design matrix (sum(M) x K_w).
- alt_idx: Integer vector of alternative indices.
- choice_idx: Integer vector of chosen alternative indices.
- M: Integer vector with number of alternatives per choice situation.
- N: Number of choice situations.
- weights: Vector of weights.
- include_outside_option: Logical flag.
- rc_correlation: Logical flag.
- alt_mapping: data.table mapping alternatives to summary statistics.
- dropped_cols: Names of columns dropped due to collinearity, if any.
- data_spec: List with column-name metadata.

Examples

```
library(data.table)
set.seed(42)
N <- 50; J <- 3
dt <- data.table(id = rep(1:N, each = J), alt = rep(1:J, N))
dt[, `:=`(x1 = rnorm(.N), w1 = rnorm(.N), w2 = rnorm(.N))]
dt[, choice := 0L]
dt[, choice := sample(c(1L, rep(0L, J - 1))), by = id]
input <- prepare_mxl_data(dt, "id", "alt", "choice", "x1", c("w1", "w2"))
str(input$X)
str(input$W)
```

prepare_nl_data	<i>Prepare inputs for nested logit estimation</i>
-----------------	---

Description

Validates inputs, builds design matrices, and constructs nest structure for nested logit estimation. Calls [prepare_mnl_data](#) internally for base data preparation, then adds nest-specific fields.

Usage

```
prepare_nl_data(
  data,
  id_col,
  alt_col,
  choice_col,
  covariate_cols,
  nest_col,
  weights = NULL,
  outside_opt_label = NULL,
  include_outside_option = FALSE
)
```

Arguments

data	Data frame containing choice data.
id_col	Name of the column identifying choice situations (individuals).
alt_col	Name of the column identifying alternatives.
choice_col	Name of the column indicating chosen alternative (1 = chosen, 0 = not chosen).
covariate_cols	Vector of names of columns to be used as covariates.
nest_col	Name of the column mapping each alternative to its nest. Every alternative must belong to exactly one nest.
weights	Optional vector of weights for each choice situation. If NULL, equal weights are used.
outside_opt_label	Label for the outside option (if any). If NULL, no outside option is assumed.
include_outside_option	Logical indicating whether to include an outside option in the model.

Value

A `choicer_data_nl` object (list) containing:

- All fields from [prepare_mnl_data](#) (`X`, `alt_idx`, `choice_idx`, `M`, `N`, `weights`, `include_outside_option`, `alt_mapping`, `dropped_cols`).
- `nest_idx`: Integer vector of length `J` mapping each alternative (in `alt_mapping` row order) to its nest.
- `data_spec`: List with column name metadata including `nest_col`.

Examples

```

library(data.table)
set.seed(42)
N <- 50; J <- 4
dt <- data.table(id = rep(1:N, each = J), alt = rep(1:J, N))
dt[, `:=`(x1 = rnorm(.N), x2 = rnorm(.N))]
dt[, nest := ifelse(alt <= 2, "A", "B")]
dt[, choice := 0L]
dt[, choice := sample(c(1L, rep(0L, J - 1))), by = id]
input <- prepare_nl_data(dt, "id", "alt", "choice", c("x1", "x2"), "nest")
input$nest_idx
input$alt_mapping

```

```
print.choicer_fit      Print a choicer_fit object
```

Description

Prints a brief summary of the fitted model.

Usage

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

Arguments

```
x          A choicer_fit object.
...        Additional arguments (ignored).
```

Value

The object invisibly.

Examples

```

library(data.table)
set.seed(42)
N <- 50; J <- 3
dt <- data.table(id = rep(1:N, each = J), alt = rep(1:J, N))
dt[, `:=`(x1 = rnorm(.N), x2 = rnorm(.N))]
dt[, choice := 0L]
dt[, choice := sample(c(1L, rep(0L, J - 1))), by = id]
fit <- run_mnlogit(dt, "id", "alt", "choice", c("x1", "x2"))
print(fit)

```

```
print.summary.choicer_mnl
      Print summary for multinomial logit model
```

Description

Print summary for multinomial logit model

Usage

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

Arguments

`x` A summary.choicer_mnl object.
`...` Additional arguments (ignored).

Value

The object invisibly.

Examples

```
library(data.table)
set.seed(42)
N <- 50; J <- 3
dt <- data.table(id = rep(1:N, each = J), alt = rep(1:J, N))
dt[, `:=`(x1 = rnorm(.N), x2 = rnorm(.N))]
dt[, choice := 0L]
dt[, choice := sample(c(1L, rep(0L, J - 1))), by = id]
fit <- run_mnlogit(dt, "id", "alt", "choice", c("x1", "x2"))
print(summary(fit))
```

```
print.summary.choicer_mxl
      Print summary for mixed logit model
```

Description

Print summary for mixed logit model

Usage

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

Arguments

x A summary.choicer_mxl object.
 ... Additional arguments (ignored).

Value

The object invisibly.

Examples

```
library(data.table)
set.seed(42)
N <- 50; J <- 3
dt <- data.table(id = rep(1:N, each = J), alt = rep(1:J, N))
dt[, `:=`(x1 = rnorm(.N), w1 = rnorm(.N))]
dt[, choice := 0L]
dt[, choice := sample(c(1L, rep(0L, J - 1))), by = id]
fit <- run_mxlogit(
  data = dt, id_col = "id", alt_col = "alt", choice_col = "choice",
  covariate_cols = "x1", random_var_cols = "w1", S = 50L
)
print(summary(fit))
```

```
print.summary.choicer_nl
```

Print summary for nested logit model

Description

Print summary for nested logit model

Usage

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

Arguments

x A summary.choicer_nl object.
 ... Additional arguments (ignored).

Value

The object invisibly.

Examples

```

library(data.table)
set.seed(42)
N <- 50; J <- 4
dt <- data.table(id = rep(1:N, each = J), alt = rep(1:J, N))
dt[, `:=`(x1 = rnorm(.N), x2 = rnorm(.N))]
dt[, nest := ifelse(alt <= 2, "A", "B")]
dt[, choice := 0L]
dt[, choice := sample(c(1L, rep(0L, J - 1))), by = id]
fit <- run_nestlogit(
  data = dt, id_col = "id", alt_col = "alt", choice_col = "choice",
  covariate_cols = c("x1", "x2"), nest_col = "nest"
)
print(summary(fit))

```

recovery_table	<i>Parameter recovery table</i>
----------------	---------------------------------

Description

Compares fitted coefficients to a set of true parameter values on the same scale as the estimator's internal parameterization. Returns one row per estimated parameter with true value, estimate, standard error, bias, relative bias (%), z-score against the truth, Wald CI, and a coverage indicator.

Usage

```

recovery_table(object, truth = NULL, level = 0.95, ...)

## S3 method for class 'choicer_fit'
recovery_table(object, truth = NULL, level = 0.95, ...)

## S3 method for class 'choicer_mc'
recovery_table(object, truth = NULL, level = 0.95, ...)

```

Arguments

object	A choicer_fit object (MNL, MXL, or NL) or a choicer_mc result.
truth	Either a choicer_sim object (whose \$true_params will be used) or a named list of true parameter values.
level	Confidence level for the Wald CI and coverage indicator. Default 0.95.
...	Unused.

Details

For MXL fits the sigma block compares the raw Cholesky parameters (L_params), not the reconstructed covariance matrix. For log-normal random-coefficient means the raw mu estimate is compared directly; callers who want recovery on the DGP scale ($\exp(\mu)$) should transform both sides before calling.

When the estimator has normalized the first inside alternative's ASC to zero (which happens for MNL/MXL with `include_outside_option = FALSE` and no outside option baked into the fit), the first entry of `truth$delta` is dropped before the comparison so lengths match.

Value

See class-specific methods.

Methods (by class)

- `recovery_table(choicer_fit)`: Returns a `choicer_recovery` object (a `data.table`) with columns `parameter`, `group`, `true`, `estimate`, `se`, `bias`, `rel_bias_pct`, `z_vs_true`, `lower_ci`, `upper_ci`, `covers`.
- `recovery_table(choicer_mc)`: For a `choicer_mc` object, delegates to `summary(object, level)` and returns a `choicer_mc_summary`. Inspect `object$replications` directly for per-rep detail.

Examples

```
sim <- simulate_mnl_data(N = 2000, J = 4, seed = 123)
fit <- run_mnlogit(
  data = sim$data, id_col = "id", alt_col = "alt", choice_col = "choice",
  covariate_cols = c("x1", "x2"),
  outside_opt_label = 0L, include_outside_option = FALSE, use_asc = TRUE
)
recovery_table(fit, sim)
```

run_mnlogit

Runs multinomial logit estimation

Description

Estimates a multinomial logit model via maximum likelihood.

Usage

```
run_mnlogit(
  data = NULL,
  id_col = NULL,
  alt_col = NULL,
  choice_col = NULL,
```

```

covariate_cols = NULL,
input_data = NULL,
optimizer = NULL,
control = list(),
weights = NULL,
outside_opt_label = NULL,
include_outside_option = FALSE,
use_asc = TRUE,
keep_data = TRUE,
nloptr_opts = NULL
)

```

Arguments

<code>data</code>	Data frame containing choice data (convenience workflow). Mutually exclusive with <code>input_data</code> .
<code>id_col</code>	Name of the column identifying choice situations (individuals).
<code>alt_col</code>	Name of the column identifying alternatives.
<code>choice_col</code>	Name of the column indicating chosen alternative (1 = chosen, 0 = not chosen).
<code>covariate_cols</code>	Vector of names of columns to be used as covariates.
<code>input_data</code>	List output from <code>prepare_mnl_data</code> (advanced workflow). Mutually exclusive with <code>data</code> .
<code>optimizer</code>	Optimizer to use: "nloptr" (default), "optim", or a custom function with signature <code>f(theta_init, eval_f, lower, upper, control)</code> where <code>eval_f(theta)</code> returns <code>list(objective, gradient)</code> . Must return a list with <code>par/value</code> (or <code>solution/objective</code>). If the custom function accepts <code>control</code> or ..., the <code>control</code> argument is forwarded; otherwise it is silently ignored.
<code>control</code>	List of optimizer-specific control parameters passed to the chosen optimizer (e.g., <code>list(maxeval = 2000)</code> for <code>nloptr</code>).
<code>weights</code>	Optional vector of weights for each choice situation. If <code>NULL</code> , equal weights are used.
<code>outside_opt_label</code>	Label for the outside option (if any). If <code>NULL</code> , no outside option is assumed.
<code>include_outside_option</code>	Logical indicating whether to include an outside option in the model.
<code>use_asc</code>	Logical indicating whether to include alternative-specific constants (ASCs) in the model.
<code>keep_data</code>	Logical. If <code>TRUE</code> (default), stores prepared data in the returned object for <code>predict()</code> and post-estimation functions.
<code>nloptr_opts</code>	Deprecated. Use <code>optimizer</code> and <code>control</code> instead.

Details

Two workflows are supported:

Convenience (default) Supply data and column names. Data preparation (`prepare_mnl_data`) is handled automatically.

Advanced Call `prepare_mnl_data` yourself and pass the result via `input_data`.

Value

A `choicer_mnl` object (inherits from `choicer_fit`). Standard S3 methods available: `summary()`, `coef()`, `vcov()`, `logLik()`, `AIC()`, `BIC()`, `nobs()`, `predict()`.

Examples

```
library(data.table)
set.seed(42)
N <- 100; J <- 3; beta_true <- c(1.0, -0.5)
dt <- data.table(id = rep(1:N, each = J), alt = rep(1:J, N))
dt[, `:=`(x1 = rnorm(.N), x2 = rnorm(.N))]
dt[, V := drop(as.matrix(.SD) %**% beta_true), .SDcols = c("x1", "x2")]
dt[, prob := exp(V) / sum(exp(V)), by = id]
dt[, choice := as.integer(alt == sample(alt, 1, prob = prob)), by = id]

fit <- run_mnlogit(dt, "id", "alt", "choice", c("x1", "x2"))
summary(fit)
coef(fit)
AIC(fit)
predict(fit, type = "shares")
```

run_mxlogit

Runs mixed logit estimation

Description

Estimates a mixed logit model via simulated maximum likelihood.

Usage

```
run_mxlogit(
  data = NULL,
  id_col = NULL,
  alt_col = NULL,
  choice_col = NULL,
  covariate_cols = NULL,
  random_var_cols = NULL,
  input_data = NULL,
  eta_draws = NULL,
  S = 100L,
  rc_dist = NULL,
  rc_mean = FALSE,
```

```

rc_correlation = FALSE,
use_asc = TRUE,
theta_init = NULL,
lower = NULL,
upper = NULL,
optimizer = NULL,
control = list(),
se_method = c("hessian", "bhhh"),
scale_vars = c("none", "sd", "mad", "iqr"),
weights = NULL,
outside_opt_label = NULL,
include_outside_option = FALSE,
keep_data = TRUE,
nloptr_opts = NULL
)

```

Arguments

data	Data frame containing choice data (convenience workflow). Mutually exclusive with input_data.
id_col	Name of the column identifying choice situations.
alt_col	Name of the column identifying alternatives.
choice_col	Name of the column indicating chosen alternative (1/0).
covariate_cols	Vector of column names for fixed covariates.
random_var_cols	Vector of column names for random coefficients.
input_data	List output from <code>prepare_mx1_data</code> (advanced workflow). Mutually exclusive with data.
eta_draws	Array of shape $K_w \times S \times N$ with standard normal draws. Required for the advanced workflow; auto-generated from S in the convenience workflow.
S	Integer number of Halton draws per individual (convenience workflow only). Default 100.
rc_dist	Integer vector indicating distribution of random coefficients (0 = normal, 1 = log-normal). Default: all normal.
rc_mean	Logical indicating whether to estimate means for random coefficients.
rc_correlation	Logical indicating whether random coefficients are correlated (convenience workflow). Ignored when input_data is used (taken from the prepared data).
use_asc	Logical indicating whether to include alternative-specific constants.
theta_init	Initial parameter vector in natural-scale units. If NULL, defaults to zeros for the β , μ , and ASC blocks, and $\log(0.5)$ on the Cholesky diagonal (so each diagonal factor $L_{pp} = 0.5$, i.e. a moderate random-coefficient variance of 0.25). The zero-on-diagonal alternative corresponds to $L_{pp} = 1$ (unit RC variance), which often lets the first L-BFGS step overshoot.
lower, upper	Optional parameter bounds for the optimizer, in natural-scale units (forward-transformed internally to scaled space when <code>scale_vars != "none"</code>). Each accepts three forms:

	NULL (default) Unbounded ($-\text{Inf}/\text{Inf}$).
	Unnamed numeric vector of length <code>n_params</code> Full-length vector ordered exactly like <code>theta_init</code> (the <code>nloptr</code> -native form).
	Named numeric vector Names must be a subset of the parameter names (β block: column names of X ; μ block: <code>Mu_<col></code> (if <code>rc_mean = TRUE</code>); Cholesky block: <code>L_<i><j></code> for $i \geq j$; ASC block: <code>ASC_<level></code>). Unlisted parameters default to $\pm\infty$. This is the recommended form for typical use, e.g. <code>lower = c(L_11 = -5, L_22 = -5)</code> to clip Cholesky diagonals.
<code>optimizer</code>	Optimizer to use: <code>"nloptr"</code> (default), <code>"optim"</code> , or a custom function. See run_mnlogit for details.
<code>control</code>	List of optimizer-specific control parameters.
<code>se_method</code>	Method for computing standard errors. Either <code>"hessian"</code> (default) for the analytical Hessian of the simulated log-likelihood, or <code>"bhhh"</code> for the BHHH/outer-product-of-gradients (OPG) estimator. BHHH scales better to large problems (many alternatives or simulation draws) but may underestimate standard errors in finite samples or away from the optimum.
<code>scale_vars</code>	Pre-estimation column scaling for design matrices. One of <code>"none"</code> (default), <code>"sd"</code> (sample standard deviation), <code>"mad"</code> (<code>stats::mad</code> , i.e. $1.4826 \times$ median absolute deviation; SD-equivalent under normality), or <code>"iqr"</code> (<code>stats::IQR(x) / 1.349</code> ; also SD-equivalent under normality). When not <code>"none"</code> , every column of X and W is divided by the chosen scale before optimization to improve Hessian conditioning. Robust scales (<code>"mad"/"iqr"</code>) better capture the bulk for heavy-tailed columns where SD is dominated by outliers, but <code>stats::mad</code> can return zero when more than half of a column's entries are identical (e.g., a sparse 0/1 dummy) and will then trigger the same near-constant-column error as <code>"sd"</code> . Coefficients and standard errors are back-transformed to the user's natural units via the delta method, so reported quantities are invariant to this choice. Columns of W associated with log-normal random coefficients (<code>rc_dist == 1</code>) are passed through unchanged, since the shifted log-normal parameterization does not admit a closed-form back-transform under multiplicative scaling.
<code>weights</code>	Optional weight vector (convenience workflow). If NULL, equal weights are used.
<code>outside_opt_label</code>	Label for the outside option (convenience workflow).
<code>include_outside_option</code>	Logical whether to include an outside option (convenience workflow).
<code>keep_data</code>	Logical. If TRUE (default), stores prepared data in the returned object for post-estimation functions.
<code>nloptr_opts</code>	Deprecated. Use <code>optimizer</code> and <code>control</code> instead.

Details

Two workflows are supported:

Convenience Supply data and column names. Data preparation ([prepare_mx1_data](#)) and Halton draw generation ([get_halton_normals](#)) are handled automatically.

Advanced Call [prepare_mx1_data](#) and [get_halton_normals](#) yourself, then pass the results via `input_data` and `eta_draws`.

Value

A `choicer_mx1` object (inherits from `choicer_fit`). Standard S3 methods available: `summary()`, `coef()`, `vcov()`, `logLik()`, `AIC()`, `BIC()`, `nobs()`.

Examples

```
library(data.table)
set.seed(42)
N <- 100; J <- 3
dt <- data.table(id = rep(1:N, each = J), alt = rep(1:J, N))
dt[, `:=`(x1 = rnorm(.N), w1 = rnorm(.N), w2 = rnorm(.N))]
dt[, choice := 0L]
dt[, choice := sample(c(1L, rep(0L, J - 1))), by = id]

fit <- run_mxlogit(
  data = dt, id_col = "id", alt_col = "alt", choice_col = "choice",
  covariate_cols = "x1", random_var_cols = c("w1", "w2"), S = 50L
)
summary(fit)
```

run_nestlogit

Runs nested logit estimation

Description

Estimates a nested logit model via maximum likelihood.

Usage

```
run_nestlogit(
  data = NULL,
  id_col = NULL,
  alt_col = NULL,
  choice_col = NULL,
  covariate_cols = NULL,
  nest_col = NULL,
  input_data = NULL,
  use_asc = TRUE,
  theta_init = NULL,
  param_names = NULL,
  optimizer = NULL,
  control = list(),
  weights = NULL,
  outside_opt_label = NULL,
  include_outside_option = FALSE,
  keep_data = TRUE,
  nloptr_opts = NULL
)
```

Arguments

data	Data frame containing choice data (convenience workflow). Mutually exclusive with input_data.
id_col	Name of the column identifying choice situations.
alt_col	Name of the column identifying alternatives.
choice_col	Name of the column indicating chosen alternative (1/0).
covariate_cols	Vector of column names for covariates.
nest_col	Name of the column mapping each alternative to its nest (convenience workflow).
input_data	List containing prepared input data for estimation (advanced workflow). Mutually exclusive with data.
use_asc	Logical indicating whether to include alternative specific constants (ASCs).
theta_init	Optional initial parameter vector. If NULL, a default vector is used.
param_names	Optional vector of parameter names. If NULL, default names are generated.
optimizer	Optimizer to use: "nloptr" (default), "optim", or a custom function. See run_mnlogit for details.
control	List of optimizer-specific control parameters.
weights	Optional weight vector (convenience workflow). If NULL, equal weights are used.
outside_opt_label	Label for the outside option (convenience workflow).
include_outside_option	Logical whether to include an outside option (convenience workflow).
keep_data	Logical. If TRUE (default), stores prepared data in the returned object for post-estimation functions.
nloptr_opts	Deprecated. Use optimizer and control instead.

Details

Two workflows are supported:

Convenience Supply data and column names (including nest_col). Data preparation ([prepare_nl_data](#)) is handled automatically.

Advanced Call [prepare_nl_data](#) (or build the input list manually) and pass it via input_data.

Value

A choicer_nl object (inherits from choicer_fit). Standard S3 methods available: summary(), coef(), vcov(), logLik(), AIC(), BIC(), nobs().

Examples

```

library(data.table)
set.seed(42)
N <- 100; J <- 4
dt <- data.table(id = rep(1:N, each = J), alt = rep(1:J, N))
dt[, `:=`(x1 = rnorm(.N), x2 = rnorm(.N))]
dt[, nest := ifelse(alt <= 2, "A", "B")]
dt[, choice := 0L]
dt[, choice := sample(c(1L, rep(0L, J - 1))), by = id]

fit <- run_nestlogit(
  data = dt, id_col = "id", alt_col = "alt", choice_col = "choice",
  covariate_cols = c("x1", "x2"), nest_col = "nest"
)
summary(fit)

```

simulate_mnl_data *Simulate multinomial logit data*

Description

Generates synthetic choice data with i.i.d. Gumbel errors, optionally with varying choice-set sizes and an outside option ($alt = 0$). Choices are determined by argmax of utility; covariates are drawn as Uniform(-1, 1).

Usage

```

simulate_mnl_data(
  N = 5000,
  J = 5,
  beta = c(0.8, -0.6),
  delta = NULL,
  seed = 123,
  outside_option = TRUE,
  vary_choice_set = TRUE
)

```

Arguments

N	Number of choice situations.
J	Number of inside alternatives.
beta	Fixed coefficients for $x_1 \dots x_{K_x}$ (length $K_x = \text{length}(\text{beta})$).
delta	Alternative-specific constants for inside alternatives (length J). Defaults to an alternating pattern of $c(0.5, -0.5)$.
seed	Random seed. Pass NULL to skip <code>set.seed()</code> (useful inside <code>monte_carlo()</code> where the caller manages RNG).

`outside_option` Logical; if TRUE (default) an outside option with `alt = 0` and zero covariates is added to every choice set.

`vary_choice_set` Logical; if TRUE (default) choice set size is sampled uniformly from $2:J$; if FALSE every individual faces all J inside alternatives.

Value

A `choicer_sim` object.

Examples

```
sim <- simulate_mnl_data(N = 1000, J = 5, seed = 123)
print(sim)
```

`simulate_mxl_data` *Simulate mixed logit data*

Description

Generates synthetic choice data with random coefficients drawn from a multivariate normal (optionally log-normal per dimension) and an additional mean shifter μ . Random coefficients are parameterized via the lower Cholesky factor of Σ . Covariates are Uniform(-1, 1) by default; columns named in `price_cols` are drawn as $-\text{Uniform}(0.1, 3)$ to mimic strictly-negative price variables.

Usage

```
simulate_mxl_data(
  N = 5000,
  J = 4,
  beta = c(0.8, -0.6),
  delta = NULL,
  mu = NULL,
  Sigma = matrix(c(1, 0.5, 0.5, 1.5), nrow = 2),
  rc_dist = NULL,
  rc_correlation = NULL,
  price_cols = NULL,
  seed = 123,
  outside_option = TRUE,
  vary_choice_set = TRUE
)
```

Arguments

N	Number of choice situations.
J	Number of inside alternatives.
beta	Fixed coefficients for $x_1 \dots x_{K_x}$ (length $K_x = \text{length}(\text{beta})$).
delta	ASCs for inside alternatives (length J). Defaults to an alternating pattern of $c(0.5, -0.5)$.
mu	Mean shifter for random coefficients (length $K_w = \text{ncol}(\text{Sigma})$). Defaults to a zero vector.
Sigma	Covariance matrix of random coefficients (square, $K_w \times K_w$).
rc_dist	Integer vector (length K_w): 0L for normal, 1L for log-normal. Default NULL is treated as all-normal.
rc_correlation	Logical; if NULL (default) it is auto-detected from the off-diagonal entries of Sigma.
price_cols	Character vector of w^* column names to draw as $-\text{Uniform}(0.1, 3)$ instead of $\text{Uniform}(-1, 1)$. Default NULL.
seed	Random seed (NULL skips <code>set.seed()</code>).
outside_option	Logical; include outside option with <code>alt = 0</code> .
vary_choice_set	Logical; if TRUE (default) choice set size is sampled uniformly from $2:J$.

Details

Random coefficients are constructed to match the estimator's parameterization in `src/mxlogit.cpp`. For every dimension the raw draw is $L \text{ %% } \eta$ where $\eta \sim N(0, I)$. A normal random coefficient (`rc_dist = 0`) is then $\gamma_k = \mu_k + (L \text{ %% } \eta)_k$. A log-normal random coefficient (`rc_dist = 1`) follows the shifted log-normal $\beta_k = \exp(\mu_k) + \exp((L \text{ %% } \eta)_k)$ – not the textbook $\exp(\mu_k + \sigma_k * \eta)$ – so μ_k in `true_params$mu` is on the same scale the estimator recovers and `recovery_table()` can compare like-for-like.

Value

A `choicer_sim` object. `true_params` includes `beta`, `delta`, `Sigma`, `L_params` (packed Cholesky parameters), `mu`, `rc_dist`, `rc_correlation`.

Examples

```
sim <- simulate_mxl_data(N = 1000, J = 4, seed = 123)
print(sim)
```

simulate_nl_data	<i>Simulate nested logit data</i>
------------------	-----------------------------------

Description

Generates synthetic choice data with nested logit probabilities computed analytically (log-sum-exp over inclusive values), then samples choices from the implied multinomial. The outside option ($j = 0$) sits in a singleton nest with $\lambda = 1$.

Usage

```
simulate_nl_data(
  N = 10000,
  beta = c(1.5, -0.8),
  delta = c(`1` = 0.5, `2` = 0.3, `3` = -0.2, `4` = -0.5, `5` = 0.4),
  nests = list(c(1, 2), c(3, 4, 5)),
  lambdas = c(0.8, 0.2),
  seed = 123
)
```

Arguments

N	Number of choice situations.
beta	Fixed coefficients for covariates X , W (length 2 by default).
delta	Named numeric vector of ASCs for inside alternatives.
nests	List of integer vectors defining nest membership for inside alternatives.
lambdas	Numeric vector of dissimilarity parameters, one per nest.
seed	Random seed (NULL skips <code>set.seed()</code>).

Value

A `choicer_sim` object. `true_params` includes `beta`, `delta`, `lambdas`; `settings` includes the `nest_structure`. The returned data retains a `nest` column (integer, with 0L for the outside option) for convenient use with `run_nestlogit()`.

Note

Unlike `simulate_mnl_data()` and `simulate_mxl_data()`, this function does not expose `outside_option` or `vary_choice_set` flags. The outside option ($j = 0$) is always present as a singleton nest with $\lambda = 1$, and every individual faces the full set of inside alternatives. Add these flags if downstream use cases need them.

Examples

```
sim <- simulate_nl_data(N = 2000, seed = 123)
print(sim)
```

summary.choicer_mnl *Summary for multinomial logit model*

Description

Computes and returns a coefficient summary table with standard errors, z-values, p-values, and significance codes. Triggers lazy Hessian computation if standard errors have not been computed yet.

Usage

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

Arguments

object A choicer_mnl object.
... Additional arguments (ignored).

Value

A summary.choicer_mnl object (list with coefficients table and metadata).

Examples

```
library(data.table)
set.seed(42)
N <- 50; J <- 3
dt <- data.table(id = rep(1:N, each = J), alt = rep(1:J, N))
dt[, `:=`(x1 = rnorm(.N), x2 = rnorm(.N))]
dt[, choice := 0L]
dt[, choice := sample(c(1L, rep(0L, J - 1))), by = id]
fit <- run_mnlogit(dt, "id", "alt", "choice", c("x1", "x2"))
summary(fit)
```

summary.choicer_mxl *Summary for mixed logit model*

Description

Computes coefficient summary with delta-method transformation for variance parameters (Cholesky to covariance scale) and log-normal mean parameters. Triggers lazy Hessian computation if standard errors have not been computed yet.

Usage

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

Arguments

```
object      A choicer_mxl object.
...         Additional arguments (ignored).
```

Value

A summary.choicer_mxl object.

Examples

```
library(data.table)
set.seed(42)
N <- 50; J <- 3
dt <- data.table(id = rep(1:N, each = J), alt = rep(1:J, N))
dt[, `:=`(x1 = rnorm(.N), w1 = rnorm(.N))]
dt[, choice := 0L]
dt[, choice := sample(c(1L, rep(0L, J - 1))), by = id]
fit <- run_mxlogit(
  data = dt, id_col = "id", alt_col = "alt", choice_col = "choice",
  covariate_cols = "x1", random_var_cols = "w1", S = 50L
)
summary(fit)
```

summary.choicer_nl *Summary for nested logit model*

Description

Triggers lazy Hessian computation if standard errors have not been computed yet.

Usage

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

Arguments

```
object      A choicer_nl object.
...         Additional arguments (ignored).
```

Value

A summary.choicer_nl object.

Examples

```

library(data.table)
set.seed(42)
N <- 50; J <- 4
dt <- data.table(id = rep(1:N, each = J), alt = rep(1:J, N))
dt[, `:=`(x1 = rnorm(.N), x2 = rnorm(.N))]
dt[, nest := ifelse(alt <= 2, "A", "B")]
dt[, choice := 0L]
dt[, choice := sample(c(1L, rep(0L, J - 1))), by = id]
fit <- run_nestlogit(
  data = dt, id_col = "id", alt_col = "alt", choice_col = "choice",
  covariate_cols = c("x1", "x2"), nest_col = "nest"
)
summary(fit)

```

vcov.choicer_fit

Extract variance-covariance matrix from a choicer_fit object

Description

Triggers lazy Hessian computation if vcov has not been computed yet.

Usage

```

## S3 method for class 'choicer_fit'
vcov(object, ...)

```

Arguments

```

object      A choicer_fit object.
...         Additional arguments (ignored).

```

Value

Named variance-covariance matrix, or NULL if unavailable.

Examples

```

library(data.table)
set.seed(42)
N <- 50; J <- 3
dt <- data.table(id = rep(1:N, each = J), alt = rep(1:J, N))
dt[, `:=`(x1 = rnorm(.N), x2 = rnorm(.N))]
dt[, choice := 0L]
dt[, choice := sample(c(1L, rep(0L, J - 1))), by = id]
fit <- run_mnlogit(dt, "id", "alt", "choice", c("x1", "x2"))
vcov(fit)

```

Index

blp, 3
blp.choicer_mnl, 4
blp.choicer_mxl, 5
blp_contraction, 6
build_var_mat, 7

coef.choicer_fit, 8

diversion_ratios, 8
diversion_ratios.choicer_mnl, 9
diversion_ratios.choicer_mxl, 10

elasticities, 11
elasticities.choicer_mnl, 12
elasticities.choicer_mxl, 12

get_halton_normals, 13, 55

jacobian_vech_Sigma, 14

logLik.choicer_fit, 15

mc_asymptotics, 15
mnl_diversion_ratios_parallel, 17
mnl_elasticities_parallel, 18
mnl_loglik_gradient_parallel, 19
mnl_loglik_hessian_parallel, 21
mnl_predict, 22
mnl_predict_shares, 23
monte_carlo, 24
monte_carlo(), 16
mxl_bhhh_parallel, 25
mxl_blp_contraction, 27
mxl_diversion_ratios_parallel, 29
mxl_elasticities_parallel, 30
mxl_hessian_parallel, 32
mxl_loglik_gradient_parallel, 33
mxl_predict, 35
mxl_predict_shares, 36

new_choicer_sim, 37

nl_loglik_gradient_parallel, 38
nl_loglik_numeric_hessian, 39
nobs.choicer_fit, 40

predict.choicer_mnl, 41
predict.choicer_mxl, 42
prepare_mnl_data, 43, 46, 52, 53
prepare_mxl_data, 44, 54, 55
prepare_nl_data, 46, 57
print.choicer_fit, 47
print.summary.choicer_mnl, 48
print.summary.choicer_mxl, 48
print.summary.choicer_nl, 49

recovery_table, 50
recovery_table(), 37
run_mnlogit, 51, 55, 57
run_mxlogit, 53
run_nestlogit, 56
run_nestlogit(), 61

simulate_mnl_data, 58
simulate_mnl_data(), 37, 61
simulate_mxl_data, 59
simulate_mxl_data(), 37, 61
simulate_nl_data, 61
simulate_nl_data(), 37
summary.choicer_mnl, 62
summary.choicer_mxl, 62
summary.choicer_nl, 63

vcov.choicer_fit, 64