Package: rmdcev (via r-universe)

October 15, 2026

Type Package

Title Kuhn-Tucker and Multiple Discrete-Continuous Extreme Value Models

Version 1.3.4

Maintainer Patrick Lloyd-Smith <patrick.lloydsmith@usask.ca>

Description Estimates and simulates Kuhn-Tucker demand models with individual heterogeneity. The package implements the multiple-discrete continuous extreme value (MDCEV) model and the Kuhn-Tucker specification common in the environmental economics literature on recreation demand. Latent class and random parameters specifications can be implemented and the models are fit using maximum likelihood estimation or Bayesian estimation. All models are implemented in 'Stan' (see Stan Development Team, 2019) <https://mc-stan.org/>. The package also implements demand forecasting (Pinjari and Bhat (2011) <https://repositories.lib.utexas.edu/handle/2152/23880>) and welfare calculation (Lloyd-Smith (2018) <doi:10.1016/j.jocm.2017.12.002>) for policy simulation. 'Stan' models can be estimated using either the 'cmdstanr' (default) or 'rstan' backend. If using 'cmdstanr', then user will need to install 'cmdstanr' manually <https://mc-stan.org/cmdstanr/>.

License MIT + file LICENSE

Depends R (>= 4.1.0), Rcpp (>= 1.0.5), methods

Imports rstan (>= 2.29.0), posterior (>= 1.0.0), rstantools (>= 2.3.0), dplyr (>= 1.0.0), purrr, tibble, tidyr, utils, stats, Formula

LinkingTo BH (>= 1.72.0), Rcpp, RcppEigen (>= 0.3.3.3.0), RcppParallel (>= 5.0.1), rstan (>= 2.29.0), StanHeaders (>= 2.29.0)

Language en-US

Encoding UTF-8

LazyData true

SystemRequirements GNU make, CmdStan (>= 2.29.0)

NeedsCompilation yes

RoxygenNote 7.3.3

Biarch true

Suggests apollo, bench, knitr, rmarkdown, testthat (>= 3.0.0), cmdstanr

VignetteBuilder knitr

Config/testthat/edition 3

Additional_repositories https://stan-dev.r-universe.dev/

URL https://github.com/plloydsmith/rmdcev

BugReports https://github.com/plloydsmith/rmdcev/issues

Author Patrick Lloyd-Smith [aut, cre], Trustees of Columbia University [cph]

Config/pak/sysreqs make

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

Date/Publication 2026-09-14 23:40:07 UTC

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

RemoteRef HEAD

RemoteSha e1e51979cb0205c554e5e15989da8288b76c3f3b

Contents

CreateBlankPolicies . . . 3
data_rec . . . 4
GenerateMDCEVData . . . 4
mdcev . . . 6
mdcev.data . . . 10
mdcev.sim . . . 11
PrepareSimulationData . . . 13
rmdcev_get_rng . . . 14
rmdcev_get_stream . . . 15
simulation_functions . . . 15

Index **18**

CreateBlankPolicies *CreateBlankPolicies*

Description

Create 'zero effect' policies that can then be modified by the user

Usage

```
CreateBlankPolicies(npols, model, price_change_only = TRUE)
```

Arguments

npols	Number of policies to simulate
model	Estimated model from mdcev
price_change_only	Logical value for whether to include policy changes to dat_psi. Defaults to TRUE. TRUE implies that only price changes are used in simulation.

Value

A named list with four elements:

price_p A list of npols numeric vectors, each of length $J + 1$ (number of non-numeraire alternatives plus the numeraire), initialised to zero. Modify these vectors to specify proportional price changes for each policy.

dat_psi_p If price_change_only = FALSE and the model has psi variables, a list of npols matrices copied from the estimated model's psi covariate data (one row per individual-alternative combination). NULL otherwise.

dat_phi_p If price_change_only = FALSE and the model is a kt_ee specification with phi variables, a list of npols matrices copied from the estimated model's phi covariate data. NULL otherwise.

price_change_only Logical; the value of the price_change_only argument, retained for use by mdcev.sim().

Examples

```
data_rec <- mdcev.data(data_rec, subset = id <= 500, id.var = "id",
                alt.var = "alt", choice = "quant")

mdcev_est <- mdcev( ~ 0, data = data_rec,
               model = "hybrid0", algorithm = "MLE",
               std_errors = "mvn", backend = "rstan")
CreateBlankPolicies(npols = 2, mdcev_est)
```

data_rec *Recreation data from Value of Nature to Canadians Survey*

Description

Data from 2000 individuals from the Value of Nature to Canadians (VNC) survey. The travel costs are calculated using the approach described in Lloyd-Smith (2020)

Usage

```
data(data_rec)
```

Format

A tibble with 34000 rows and 8 variables

Source

Canadian Nature Survey 2012

References

Federal, Provincial, and Territorial Governments of Canada. 2014. "2012 Canadian Nature Survey: Awareness, Participation, and Expenditures in Nature-Based Recreation, Conservation, and Subsistence Activities." Ottawa, ON: Canadian Councils of Resource Ministers. (catalogue no. B64-513/1-2012E-PDF)

Lloyd-Smith, P (2022). "The Economic Benefits of Recreation in Canada". Canadian Journal of Economics. doi:10.1111/caje.12560

GenerateMDCEVData *GenerateMDCEVData*

Description

Simulate data for KT models

Usage

```
GenerateMDCEVData(
  model,
  nobs = 1000,
  nalts = 10,
  income = stats::runif(nobs, 1e+05, 150000),
  price = matrix(stats::runif(nobs * nalts, 100, 500), nobs, nalts),
  alpha_parms = 0.5,
  scale_parms = 1,
```

```
  gamma_parms = stats::runif(nalts, 1, 10),
  psi_i_parms = c(-1.5, 2, -1),
  psi_j_parms = c(-5, 0.5, 2),
  phi_parms = c(-5, 0.5, 2),
  dat_psi_i = matrix(2 * stats::runif(nobs * length(psi_i_parms)), nobs,
    length(psi_i_parms)),
 dat_psi_j = cbind(matrix(stats::runif(nalts * (length(psi_j_parms)), 0, 1), nrow =
    nalts)),
 dat_phi = cbind(matrix(stats::runif(nalts * (length(phi_parms)), 0, 1), nrow = nalts)),
  nerrs = 1,
  tol = 1e-20,
  max_loop = 999
)
```

Arguments

model	A string indicating which model specification is estimated. The options are "alpha", "gamma", "hybrid", "hybrid0", and "gamma1" for the MDCEV model and "kt_ee" for the environmental economics Kuhn-Tucker specification. The "gamma1" model fixes alpha_0 = 1 (no income effects).
nobs	Number of individuals
nalts	Number of non-numeraire alts
income	Vector of individual income
price	Matrix of prices for non-numeraire alternatives.
alpha_parms	Parameter value for alpha term
scale_parms	Parameter value for scale term
gamma_parms	Parameter value for gamma terms
psi_i_parms	Parameter value for psi terms that vary by individual
psi_j_parms	Parameter value for psi terms that vary by alt (all models except kt_ee)
phi_parms	Parameter value for phi terms that vary by alt (kt_ee model only)
dat_psi_i	(nobs X # psi_i_parms) matrix with individual-specific characteristics
dat_psi_j	(nalts X # psi_j_parms) matrix with alternative-specific variables (all models except kt_ee)
dat_phi	(nalts X # phi_parms) matrix with alternative-specific variables (kt_ee model only)
nerrs	Number of error draws for demand simulation
tol	Tolerance level for simulations if using general approach
max_loop	maximum number of loops for simulations if using general approach

Value

A ‘mdcev.data‘ object, which is a ‘data.frame‘ in long format. Also includes parms_true with parameter values

Examples

```
data <- GenerateMDCEVData(model = "gamma")
```

---

mdcev *mdcev*

Description

Fit a MDCEV model using MLE or Bayes

Usage

```
mdcev(
  formula = NULL,
  data,
  weights = NULL,
  model = c("alpha", "gamma", "hybrid", "hybrid0", "kt_ee", "gamma1"),
  n_classes = 1,
  fixed_scale1 = FALSE,
  single_scale = FALSE,
  trunc_data = FALSE,
  psi_ascs = NULL,
  gamma_ascs = TRUE,
  seed = 123L,
  max_iterations = 2000,
  jacobian_analytical_grad = TRUE,
  initial.parameters = "random",
  hessian = TRUE,
  algorithm = c("MLE", "Bayes"),
  flat_priors = NULL,
  print_iterations = TRUE,
  prior_psi_sd = 10,
  prior_gamma_sd = 10,
  prior_phi_sd = 10,
  prior_alpha_shape = 1,
  prior_scale_sd = 1,
  prior_delta_sd = 10,
  gamma_nonrandom = FALSE,
  alpha_nonrandom = FALSE,
  psi_random = NULL,
  std_errors = "deltamethod",
  n_draws = 50,
  keep_loglik = FALSE,
  random_parameters = "fixed",
  show_stan_warnings = TRUE,
  n_iterations = 200,
```

```
  n_chains = 4,
  n_cores = 4,
  max_tree_depth = 10,
  adapt_delta = 0.8,
  lkj_shape_prior = 4,
  backend = "cmdstanr",
  ...
)

## S3 method for class 'mdcev'
print(
  x,
  digits = max(3, getOption("digits") - 3),
  width = getOption("width"),
  ...
)

## S3 method for class 'mdcev'
summary(object, printCI = FALSE, ...)

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

Arguments

formula	Formula for the model to be estimated. The formula is divided in three parts, separated by the symbol |. The first part is reserved for alternative-specific and individual-specific variables in the psi parameters. Note that alternative-specific constants are handled by the psi_ascs argument. The second part corresponds for individual-specific variables that enter in the probability assignment in models with latent classes. The third part is reserved for the q_k variables included in the phi_k parameters in the KT model specification used in environmental economics model = "kt_ee".
data	The (IxJ) data to be passed to Stan of class mdcev.data including 1) id, 2) alt, 3) choice, 4) price, 5) income, and columns for alternative-specific and individual specific variables. Note: I is number of individuals and J is number of non-numeraire alternatives.
weights	an optional vector of weights. Default to 1.
model	A string indicating which model specification is estimated. The options are "alpha", "gamma", "hybrid", "hybrid0", and "gamma1" for the MDCEV model and "kt_ee" for the environmental economics Kuhn-Tucker specification. The "gamma1" model fixes alpha_0 = 1 (no income effects).
n_classes	The number of latent classes. Note that the LC model is automatically estimated as long as the prespecified number of classes is set greater than 1.
fixed_scale1	Whether to fix scale at 1.
single_scale	For lc models, whether to estimate a single scale parameter

`trunc_data` Whether the estimation should be adjusted for truncation of non-numeraire alternatives. This option is useful if the data only includes individuals with positive non-numeraire consumption levels such as recreation data collected on-site. To account for the truncation of consumption, the likelihood is normalized by one minus the likelihood of observing zero consumption for all non-numeraire alternatives (i.e. likelihood of positive consumption) following Englin, Boxall and Watson (1998) and von Haefen (2003).

`psi_ascs` Whether to include alternative-specific psi parameters. The first alternative is used as the reference category. Only specify to 1 for MDCEV models.

`gamma_ascs` Indicator to include alternative-specific gammas parameters.

`seed` Random seed.

`max_iterations` Maximum number of iterations in MLE.

`jacobian_analytical_grad`
indicator whether to use analytical gradient method for Jacobian (=1) or numerical gradient method (=0). For "kt_ee" model only,

`initial.parameters`
The default for fixed and random parameter specifications is to use random starting values. (except for the scale parameter with a starting value set to 1). For LC models, the default is to use slightly adjusted MLE point estimates from the single class model. Initial parameter values should be included in a named list. For example, the LC "hybrid" specification initial parameters can be specified as: initial.parameters = list(psi = array(0, dim = c(K, num_psi)), gamma = array(1, dim = c(K, num_alt)), alpha = array(0.5, dim = c(K, 0)), scale = array(1, dim = c(K))) where K is the number of classes (i.e. K = 1 is used for single class models), num_psi is number of psi parameters, and num_alt is number of non-numeraire alternatives.

`hessian` Whether to keep the Hessian matrix

`algorithm` Either "Bayes" for Bayes or "MLE" for maximum likelihood estimation.

`flat_priors` indicator if completely uninformative priors should be specified. Defaults to 1 if MLE used and 0 if Bayes used. If using MLE and set flat_priors = 0, penalized MLE is used and the optimizing objective is augmented with the priors.

`print_iterations`
Whether to print iteration information

`prior_psi_sd` standard deviation for normal prior with mean 0.

`prior_gamma_sd` standard deviation for half-normal prior with mean 1.

`prior_phi_sd` standard deviation for normal prior with mean 0.

`prior_alpha_shape`
shape parameter for beta distribution.

`prior_scale_sd` standard deviation for half-normal prior with mean 0.

`prior_delta_sd` standard deviation for normal prior with mean 0.

`gamma_nonrandom`
Logical. If `TRUE`, gamma parameters are not random (no individual-specific standard deviation).

alpha_nonrandom
: Logical. If TRUE, alpha parameters are not random (no individual-specific standard deviation).

psi_random
: A one-sided formula specifying which psi formula terms should be treated as random coefficients (i.e. have individual-specific draws with estimated population mean and standard deviation). Only applicable when algorithm = "Bayes" and random_parameters is "uncorr" or "corr". When NULL (default), all formula terms are random (existing behaviour). Alternative-specific constants (from psi_ascs = 1) are always random regardless of this argument. Formula terms not listed in psi_random become fixed (pooled) point estimates. Example: if the psi formula is ~ quality + income and you specify psi_random = ~ quality, then quality gets an individual-specific coefficient while income is estimated as a single pooled parameter.

std_errors
: Compute standard errors using the delta method ("deltamethod") or multivariate normal draws ("mvn"). The default is "deltamethod". Note that mvn parameter draws should be used to incorporate parameter uncertainty for demand and welfare simulation. For maximum likelihood estimation only.

n_draws
: The number of multivariate normal draws for standard error calculations if "mvn" is specified.

keep_loglik
: Whether to keep the log_lik calculations

random_parameters
: The form of the covariance matrix for Bayes. Can be 'fixed' for no random parameters, 'uncorr' for uncorrelated random parameters, or 'corr' for correlated random parameters.

show_stan_warnings
: Whether to show warnings from Stan.

n_iterations
: The number of iterations to use in Bayesian estimation. The default is for the number of iterations to be split evenly between warmup and posterior draws. The number of warmup draws can be directly controlled using the warmup argument (see rstan::sampling).

n_chains
: The number of independent Markov chains in Bayesian estimation.

n_cores
: The number of cores used to execute the Markov chains in parallel in Bayesian estimation. Can set using options(mc.cores = parallel::detectCores()).

max_tree_depth
: https://mc-stan.org/misc/warnings.html#maximum-treedepth-exceeded

adapt_delta
: https://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup

lkj_shape_prior
: Prior for Cholesky matrix

backend
: Estimation backend. Either "cmdstanr" (default) or "rstan". The cmdstanr backend reads .stan files from inst/stan/ at runtime. The rstan backend uses pre-compiled C++ in src/stanExports_*.h.

...
: Additional parameters to pass on to rstan::stan and rstan::sampling.

x, object
: an object of class 'mdcev'

digits
: the number of digits,

width
: the width of the printing,

printCI
: set to TRUE to print 95% confidence intervals

Value

A object of class mdcev

Examples

```
data(data_rec, package = "rmdcev")

data_rec <- mdcev.data(data_rec, subset = id <= 500, id.var = "id",
                alt.var = "alt", choice = "quant")

mdcev_est <- mdcev( ~ 0,
data = data_rec,
model = "hybrid0",
algorithm = "MLE",
backend = "rstan")
```

---

mdcev.data *data.frame for mdcev model*

Description

shape a ‘data.frame‘ in a suitable form for the use of the ‘mdcev‘ function and complete some data checks

Usage

```
mdcev.data(
  data,
  id.var = "id",
  alt.var = NULL,
  choice = "choice",
  price = "price",
  income = "income",
  alt.levels = NULL,
  drop.index = FALSE,
  subset = NULL,
  ...
)
```

Arguments

data	a ‘data.frame‘,
id.var	the name of the variable that contains the individual index.
alt.var	the name of the variable that contains the alternative index or the name under which the alternative index will be stored (the default name is ‘alt‘),

choice	the variable indicating the consumption of non-numeraire alternatives that is made: it has to be a numerical vector. Default is "choice".
price	the variable indicating the price of the non-numeraire alternatives. Default is "price"
income	the variable indicating the income of the individual. Default is "income". Set to NULL when using the "gamma1" model, which has no income effects and does not require income data.
alt.levels	the name of the alternatives: if null, they are guessed from the 'alt.var' argument,
drop.index	should the index variables be dropped from the 'data.frame',
subset	a logical expression which defines the subset of observations to be selected,
...	further arguments.

Value

A 'mdcev.data' object, which is a 'data.frame' in long format, *i.e.* one line for each alternative. It has a 'index' attribute, which is a 'data.frame' that contains the index of the individual ('id') and the index of the alternative ('alt').

---

mdcev.sim *mdcev.sim*

Description

Simulate welfare or demand for MDCEV model

Usage

```
mdcev.sim(
  df_indiv,
  df_common,
  sim_options,
  sim_type = c("welfare", "demand"),
  nerrs = 30,
  cond_error = TRUE,
  draw_mlhs = TRUE,
  algo_gen = NULL,
  tol = 1e-20,
  max_loop = 999,
  suppressTime = FALSE,
  stan_seed = 3,
  ...
)

## S3 method for class 'mdcev.sim'
print(
```

```
  x,
  digits = max(3, getOption("digits") - 3),
  width = getOption("width"),
  ...
)

## S3 method for class 'mdcev.sim'
summary(object, ci = 0.95, ...)

## S3 method for class 'summary.mdcev.sim'
print(
  x,
  digits = max(3, getOption("digits") - 2),
  width = getOption("width"),
  ...
)
```

Arguments

df_indiv	Prepared individual level data from PrepareSimulationData
df_common	Prepared common data from PrepareSimulationData
sim_options	Prepared simulation options from PrepareSimulationData
sim_type	Either "welfare" or "demand"
nerrs	Number of error draws for welfare analysis
cond_error	Choose whether to draw errors conditional on actual demand or not. Conditional error draws (=1) or unconditional error draws.
draw_mlhs	Generate draws using Modified Latin Hypercube Sampling algorithm (=1) or uniform (=0)
algo_gen	Type of algorithm for simulation. algo_gen = 0 for Hybrid Approach (i.e. constant alphas, only hybrid models) algo_gen = 1 for General approach (i.e. heterogeneous alpha's, all models)
tol	Tolerance level for simulations if using general approach
max_loop	maximum number of loops for simulations if using general approach
suppressTime	Suppress simulation time calculation
stan_seed	Seed for pseudo-random number generator get_rng see help(get_rng, package = "rstan")
...	Additional parameters to pass to mdcev.sim
x, object	an object of class 'mdcev.sim'
digits	the number of digits,
width	the width of the printing,
ci	choose confidence interval for simulations. Default is 95 percent.

Value

a object of class mdcev.sim which contains a list for each individual holding either 1) nsims x npols matrix of welfare changes if welfare is being simulated or 2) nsims number of lists of npols x # alternatives matrix of Marshallian demands is demand is being simulated.

See Also

[mdcev()] for the estimation of mdcev models.

Examples

```
data(data_rec, package = "rmdcev")

data_rec <- mdcev.data(data_rec, subset = id <= 500, id.var = "id",
                alt.var = "alt", choice = "quant")

mdcev_est <- mdcev( ~ 0, data = data_rec,
               model = "hybrid0", algorithm = "MLE",
               std_errors = "mvn", backend = "rstan")

policies <- CreateBlankPolicies(npols = 2,
             mdcev_est,
             price_change_only = TRUE)

df_sim <- PrepareSimulationData(mdcev_est, policies)

wtp <- mdcev.sim(df_sim$df_indiv,
                 df_common = df_sim$df_common,
                 sim_options = df_sim$sim_options,
                 cond_err = 1, nerrs = 5, sim_type = "welfare")
```

PrepareSimulationData *PrepareSimulationData*

Description

Prepare data for WTP/demand simulation from a fitted mdcev object.

Usage

```
PrepareSimulationData(object, policies, nsims = 30, class = "class1")
```

Arguments

object	An object of class mdcev.
policies	A list produced by CreateBlankPolicies containing price_p (additive price changes) and optionally dat_psi_p / dat_phi_p (alternative-attribute changes).
nsims	Number of parameter draws to use for uncertainty quantification.
class	Class label for Latent Class models (e.g. "class1").

Value

A list with `df_indiv` (individual-level data), `df_common` (shared simulation inputs), and `sim_options` (model metadata).

Examples

```
data(data_rec, package = "rmdcev")

data_rec <- mdcev.data(data_rec, subset = id <= 500, id.var = "id",
                alt.var = "alt", choice = "quant")

mdcev_est <- mdcev( ~ 0, data = data_rec,
               model = "hybrid0", algorithm = "MLE",
               std_errors = "mvn", backend = "rstan")

policies <- CreateBlankPolicies(npols = 2, mdcev_est,
             price_change_only = TRUE)

df_sim <- PrepareSimulationData(mdcev_est, policies)
```

`rmdcev_get_rng` *rmdcev_get_rng*

Description

Create a boost::ecuyer1988 RNG seeded with `seed`, returned as an external pointer suitable for passing to the compiled Stan simulation functions. Defined here (not in RcppExports.R) so it survives `Rcpp::compileAttributes()` regeneration.

Usage

```
rmdcev_get_rng(seed = 0L)
```

Arguments

`seed`	Integer seed value (default 0L).

rmdcev_get_stream *rmdcev_get_stream*

Description

Return an external pointer to `Rcpp::Rcout` for use as the `pstream__` argument in compiled Stan simulation functions. Defined here (not in RcppExports.R) so it survives `Rcpp::compileAttributes()` regeneration.

Usage

```
rmdcev_get_stream()
```

simulation_functions *Low-level MDCEV simulation functions*

Description

Compiled C++ routines that underlie `mdcev.sim` and `GenerateMDCEVData`. These functions are exported for advanced users and test code; most users should call the higher-level wrappers instead.

`DrawError_rng` Draw preference error terms for one individual using the MLHS or independent draw approach.

`MarshallianDemand` Compute Marshallian (uncompensated) demand for one individual given marginal utilities at zero.

`HicksianDemand` Compute Hicksian (compensated) demand for one individual given a reference utility level.

`ComputeUtilJ` Evaluate the KT utility function for one individual.

`CalcWTP_rng` Simulate welfare (WTP) across `nerrs` error draws for one individual under a price policy.

`CalcMarshallianDemand_rng` Simulate Marshallian demand across `nerrs` error draws for one individual under a price policy.

Usage

```
DrawError_rng(quant_num, quant_j, price_j, psi_j, phi_j, gamma_j,
              alpha, scale, model_num, nalts, nerrs, cond_error,
              draw_mlhs, base_rng__, pstream__)

MarshallianDemand(income, price, MUzero, phi, gamma, alpha, nalts,
                  algo_gen, model_num, tol_e, max_loop, pstream__)

HicksianDemand(util, price, MUzero, phi, gamma, alpha, nalts,
               algo_gen, model_num, tol_l, max_loop, pstream__)
```

```
ComputeUtilJ(income, quant_j, price_j, psi, phi_j, gamma_j, alpha,
             nalts, model_num, pstream__)

CalcWTP_rng(income, quant_j, price, price_p_policy, psi_p_sims,
            phi_p_sims, psi_sims, phi_sims, gamma_sims, alpha_sims,
            scale_sims, nerrs, cond_error, draw_mlhs, algo_gen,
            model_num, price_change_only, tol, max_loop, base_rng__,
            pstream__)

CalcMarshallianDemand_rng(income, quant_j, price, price_p_policy,
                          psi_p_sims, phi_p_sims, psi_sims, phi_sims,
                          gamma_sims, alpha_sims, scale_sims, nerrs,
                          cond_error, draw_mlhs, algo_gen, model_num,
                          price_change_only, tol, max_loop, base_rng__,
                          pstream__)
```

Arguments

base_rng__	External pointer to the Boost RNG created by rmdcev_get_rng.
pstream__	External pointer to the output stream created by rmdcev_get_stream.
model_num	Integer model type: 1 = gamma, 2 = alpha, 3 = hybrid, 4 = hybrid0, 5 = kt_ee, 6 = gamma1.
income	Individual income (scalar).
quant_j	Vector of non-numeraire quantities.
quant_num	Numeraire quantity consumed.
price_j	Vector of non-numeraire prices.
price	Full price vector (numeraire first).
price_p_policy	Matrix of counterfactual prices, one row per policy scenario.
psi_j, psi	Psi utility parameters.
phi_j, phi	Phi parameters (kt_ee model).
gamma_j, gamma	Gamma satiation parameters.
alpha	Alpha parameters.
scale	Scale parameter.
psi_sims, phi_sims, gamma_sims, alpha_sims, scale_sims	Lists of parameter draws, one element per simulation, for the baseline scenario.
psi_p_sims, phi_p_sims	Lists of psi and phi parameter draws for the policy scenarios.
nalts	Number of non-numeraire alternatives.
nerrs	Number of error draws.
cond_error	1 = conditional error draws; 0 = unconditional.
draw_mlhs	1 = Modified Latin Hypercube Sampling; 0 = iid.
algo_gen	Algorithm: 0 = hybrid (fast, requires alpha = 0); 1 = general bisection.

price_change_only
: 1 = only prices change across policies, so the baseline psi and phi draws are reused; 0 = psi and phi also change across policies.

MUzero
: Vector of marginal utilities at zero consumption.

util
: Reference utility level (Hicksian demand).

tol, tol_e, tol_l
: Convergence tolerance for bisection.

max_loop
: Maximum bisection iterations.

Value

DrawError_rng returns a matrix of error draws with nerrs rows and nalts + 1 columns.

MarshallianDemand and HicksianDemand return a numeric vector of length nalts + 1 holding the numeraire quantity followed by the non-numeraire quantities.

ComputeUtilJ returns a scalar utility value.

CalcWTP_rng and CalcMarshallianDemand_rng return a list with one element per policy scenario, each holding the simulated welfare measure or demand averaged over the nerrs error draws.

See Also

mdcev.sim, PrepareSimulationData, rmdcev_get_rng, rmdcev_get_stream

Index

∗ **datasets**
data_rec, 4

CalcMarshallianDemand_rng *(*simulation_functions*)*, 15
CalcWTP_rng *(*simulation_functions*)*, 15
ComputeUtilJ *(*simulation_functions*)*, 15
CreateBlankPolicies, 3, *13*

data_rec, 4
DrawError_rng *(*simulation_functions*)*, 15

GenerateMDCEVData, 4, *15*

HicksianDemand *(*simulation_functions*)*, 15

MarshallianDemand *(*simulation_functions*)*, 15
mdcev, 6
mdcev.data, *7*, 10
mdcev.sim, 11, *15*, *17*

PrepareSimulationData, 13, *17*
print.mdcev *(*mdcev*)*, 6
print.mdcev.sim *(*mdcev.sim*)*, 11
print.summary.mdcev *(*mdcev*)*, 6
print.summary.mdcev.sim *(*mdcev.sim*)*, 11

rmdcev_get_rng, 14, *16*, *17*
rmdcev_get_stream, 15, *16*, *17*

simulation_functions, 15
summary.mdcev *(*mdcev*)*, 6
summary.mdcev.sim *(*mdcev.sim*)*, 11