Package 'rmdcev'

November 22, 2019

Type Package

Title Multiple Discrete-Continuous Extreme Value (MDCEV) Model

Version 1.1.1

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

Description Estimates different multiple discrete-continuous extreme value (MDCEV) demand model specifications with observed and unobserved individual heterogeneity (Bhat (2008) <doi:10.1016/j.trb.2007.06.002>). Fixed parameter, latent class, and random parameter models can be estimated. These models are estimated using maximum likelihood or Bayesian estimation techniques and are implemented in 'Stan', which is a C++ package for performing full Bayesian inference (see Stan Development Team (2018) <http://mc-stan.org>). The 'rmdcev' package also includes functions for demand simulation (Pinjari and Bhat (2011) <https://repositories.lib.utexas.edu/handle/2152/23880>) and welfare simulation (Lloyd-Smith (2018) <doi:10.1016/j.jocm.2017.12.002>).

License MIT + file LICENSE

Depends R (>= 3.5.0), Rcpp (>= 0.12.0), methods

Imports rstan (>= 2.18.2), rstantools (>= 1.5.1), dplyr (>= 0.7.8), tidyselect, parallel, stringr, purrr, tibble, tidyr, rlang, utils, stats, tmvtnorm, Formula

LinkingTo BH (>= 1.66.0), Rcpp (>= 0.12.0), RcppEigen (>= 0.3.3.4.0), rstan (>= 2.18.2), StanHeaders (>= 2.18.0)

Encoding UTF-8

LazyData true

NeedsCompilation yes

Repository CRAN

SystemRequirements GNU make C++14

RoxygenNote 7.0.0

Suggests knitr, rmarkdown, testthat

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

Date/Publication 2019-11-22 17:50:02 UTC

R topics documented:

`CreateBlankPolicies` *CreateBlankPolicies*

Description

Create 'zero effect' policies that can be modified

Usage

```
CreateBlankPolicies(npols, nalts, dat_psi, price_change_only)
```

Arguments

`npols`	Number of policies to simulate
`nalts`	Number of non-numeraire alts
`dat_psi`	Psi data matrix used in estimation
`price_change_only`	Logical value for whether to include policy changes to dat_psi. TRUE implies that only price changes are used in simulation.

Examples

```
CreateBlankPolicies(npols = 2, nalts = 10, dat_psi = NULL, price_change_only = TRUE)
```

CreateListsCol *CreateListsCol*

Description

Convert matrix x to a list with each row as an element

Usage

```
CreateListsCol(x)
```

Arguments

x	matrix to be converted to list

Examples

```
tmp <- matrix(0, nrow = 10, ncol = 5)
tmp_list <- CreateListsCol(tmp)
```

CreateListsRow *CreateListsRow*

Description

Convert matrix x to a list with each row as an element

Usage

```
CreateListsRow(x)
```

Arguments

x	matrix to be converted to list

Value

A list

Examples

```
tmp <- matrix(0, nrow = 10, ncol = 5)
tmp_list <- CreateListsRow(tmp)
```

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

Description

Data from 997 individuals from the Value of Nature to Canadians (VNC) survey. The travel costs are calculated using the approach described in Lloyd-Smith (2019)

Usage

```
data(data_rec)
```

Format

A tibble with 16949 rows and 9 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. (PubMed)

Lloyd-Smith, P (2019). "The Economic Benefits of Outdoor Recreation in Canada". Working Paper

GenClassNames *GenClassNames*

Description

Create names for LC

Usage

```
GenClassNames(parms_names, n_classes)
```

Arguments

parms_names	list of parameter names
n_classes	The number of latent classes.

Value

A vector of LC names

GenerateMDCEVData *GenerateMDCEVData*

Description

Simulate data for MDCEV model

Usage

```
GenerateMDCEVData(
  model,
  nobs = 1000,
  nalts = 10,
  inc_lo = 1e+05,
  inc_hi = 150000,
  price_lo = 100,
  price_hi = 500,
  alpha_parms = 0.5,
  scale_parms = 1,
  gamma_parms = stats::runif(nalts, 1, 2),
  psi_i_parms = c(-1.5, 3, -2, 1, 2),
  psi_j_parms = c(-5, 0.5, 2),
  nerrs = 1,
  tol = 1e-20,
  max_loop = 999
)
```

Arguments

model	A string indicating which model specification is estimated. The options are "alpha", "gamma", "hybrid" and "hybrid0".
nobs	Number of individuals
nalts	Number of non-numeraire alts
inc_lo	Low bound of income for uniform draw
inc_hi	High bound of income for uniform draw
price_lo	Low bound of price for uniform draw
price_hi	High bound of price for uniform draw
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
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 = "hybrid0")
```

GenerateMDCEVDataRP *GenerateMDCEVDataRP*

Description

Simulate random parameter data for MDCEV model

Usage

```
GenerateMDCEVDataRP(
  model,
  nobs = 1000,
  nalts = 10,
  inc_lo = 1e+05,
  inc_hi = 150000,
  price_lo = 100,
  price_hi = 500,
  alpha_parms = 0.5,
  scale_parms = 1,
  gamma_parms = stats::runif(nalts, 1, 10),
  psi_j_parms = c(-5, 0.5, 2),
  nerrs = 1,
  tol = 1e-20,
  max_loop = 999,
  corr = 0
)
```

Arguments

model	A string indicating which model specification is estimated. The options are "alpha", "gamma", "hybrid" and "hybrid0".
nobs	Number of individuals
nalts	Number of non-numeraire alts
inc_lo	Low bound of income for uniform draw
inc_hi	High bound of income for uniform draw
price_lo	Low bound of price for uniform draw

price_hi	High bound of price for uniform draw
alpha_parms	Parameter value for alpha term
scale_parms	Parameter value for scale term
gamma_parms	Parameter value for gamma terms
psi_j_parms	Parameter value for psi terms that vary by alt
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
corr	Whether to draw correlated random parameters (=1) or uncorrelated (=0)

Value

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

Examples

```
data <- GenerateMDCEVDataRP(model = "hybrid0")
```

---

maxlikeMDCEV *maxlikeMDCEV*

Description

Fit a MDCEV model with MLE

Usage

```
maxlikeMDCEV(stan_data, initial.parameters, mle_options, ...)
```

Arguments

stan_data	data for model fromatted from processMDCEVdata
initial.parameters	Specify initial parameters intead of starting at random. Initial parameter values should be included in a named list. For the "hybrid" specification, initial parameters can be specified as: init = list(psi = array(0, dim = c(1, num_psi)), gamma = array(1, dim = c(1, num_alt)), alpha = array(0.5, dim = c(1, 0)), scale = array(1, dim = c(1))) where num_psi is number of psi parameters and num_alt is number of non-numeraire alternatives
mle_options	modeling options for MLE
...	Additional parameters to pass on to rstan::stan and rstan::sampling.

mdcev *mdcev*

Description

Fit a MDCEV model using MLE or Bayes

Usage

```
mdcev(
  formula = NULL,
  data,
  weights = NULL,
  model = c("alpha", "gamma", "hybrid", "hybrid0"),
  n_classes = 1,
  fixed_scale1 = 0,
  trunc_data = 0,
  seed = "123",
  max_iterations = 2000,
  initial.parameters = NULL,
  algorithm = c("MLE", "Bayes"),
  flat_priors = NULL,
  print_iterations = TRUE,
  hessian = TRUE,
  prior_psi_sd = 10,
  prior_gamma_sd = 10,
  prior_alpha_sd = 0.5,
  prior_scale_sd = 1,
  prior_delta_sd = 10,
  gamma_fixed = 0,
  alpha_fixed = 0,
  std_errors = "mvn",
  n_draws = 50,
  keep_loglik = 0,
  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,
  ...
)

## 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 two parts, separated by the symbol |. The first part is reserved for variables in the psi parameter. These can include alternative-specific and individual-specific variables. The second part corresponds for individual-specific variables that enter in the probability assignment in models with latent classes.
data	The (IxJ) data to be passed to Stan of class mdcev.data including 1) id, 2) alt, 3) quant, 4) price, 5) income, and columns for psi variables. Arrange data by id then alt. 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" and "hybrid0".
n_classes	The number of latent classes.
fixed_scale1	Whether to fix scale at 1.
trunc_data	Whether the estimation should be adjusted for truncation
seed	Random seed.
max_iterations	Maximum number of iterations in MLE estimation.
initial.parameters	Specify initial parameters intead of starting at random. Initial parameter values should be included in a named list. For the "hybrid" specification, initial parameters can be specified as: init = list(psi = array(0, dim = c(1, num_psi)), gamma = array(1, dim = c(1, num_alt)), alpha = array(0.5, dim = c(1, 0)), scale = array(1, dim = c(1))) where num_psi is number of psi parameters and num_alt is number of non-numeraire alternatives
algorithm	Either "Bayes" for Bayes or "MLE" for maximum liklihood estimation.
flat_priors	indicator if completely uninformative priors should be specified. If using MLE, the optimizing function will then be equal to log-likelihood. Defaults to 1 if MLE used and 0 if Bayes used.
print_iterations	Whether to print iteration information
hessian	Wheter to keep the Hessian matrix

prior_psi_sd	standard deviation for normal prior with mean 0.
prior_gamma_sd	standard deviation for normal prior with mean 0.
prior_alpha_sd	standard deviation for normal prior with mean 0.5.
prior_scale_sd	standard deviation for normal prior with mean 1.
prior_delta_sd	standard deviation for normal prior with mean 0.
gamma_fixed	indicator if gamma parameters should be fixed (i.e. not random).
alpha_fixed	indicator if alpha parameters should be fixed (i.e. not random).
std_errors	Compute standard errors using the delta method ("deltamethod") or multivariate normal draws ("mvn"). The default is "mvn" as only mvn parameter draws are required for demand and welfare simulation.
n_draws	The number of multivariate normal draws for standard error calculations.
keep_loglik	Whether to keep the log_lik calculations
random_parameters	The form of the covariance matrix for Bayes. Can be 'fixed', 'uncorr, 'corr'.
show_stan_warnings	Whether to show warnings from Stan.
n_iterations	The number of iterations in Bayesian estimation.
n_chains	The number of chains in Bayesian estimation.
n_cores	The number of cores to use in Bayesian estimation. Can set using options(mc.cores = parallel::detectCores()).
max_tree_depth	http://mc-stan.org/misc/warnings.html#maximum-treedepth-exceeded
adapt_delta	http://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup
lkj_shape_prior	Prior for Cholesky matrix
...	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,
                alt.var = "alt", choice = "quant")

mdcev_est <- mdcev( ~ 1,
```

```
data = data_rec,
model = "hybrid0",
algorithm = "MLE")
```

`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 = NULL,
  alt.var = NULL,
  choice = NULL,
  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
`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".
`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 = 1,
  draw_mlhs = 1,
  algo_gen = NULL,
  tol = 1e-20,
  max_loop = 999,
  suppressTime = FALSE,
  ...
)

## 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 algorhitm for simulation. algo_gen = 0 for Hybrid Approach (i.e. constant alphas, only model 3/4) alog_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	Supress simulation time calculation
...	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,
                alt.var = "alt", choice = "quant")

mdcev_est <- mdcev( ~ 1, data = data_rec,
               model = "hybrid0", algorithm = "MLE")

policies <- CreateBlankPolicies(npols = 2,
nalts = mdcev_est[["stan_data"]][["J"]],
```

```
dat_psi = mdcev_est[["stan_data"]][["dat_psi"]],
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 simulation

Usage

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

Arguments

`object`	an object of class 'mdcev'
`policies`	list containing price_p with additive price increases, and dat_psi_p with new psi data
`nsims`	Number of simulation draws to use for parameter uncertainty
`class`	The class number for Latent Class models.

Value

A list with individual-specific data (df_indiv) and common data (df_common) and n_classes for number of classes and model_num for model type

Examples

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

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

mdcev_est <- mdcev( ~ 1,
data = data_rec,
model = "hybrid0",
algorithm = "MLE")
```

```
policies <- CreateBlankPolicies(npols = 2,
nalts = mdcev_est[["stan_data"]][["J"]],
dat_psi = mdcev_est[["stan_data"]][["dat_psi"]],
price_change_only = TRUE)

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

`ReduceStanFitSize` *ReduceStanFitSize*

Description

This function reduces the size of the stan.fit object

Usage

```
ReduceStanFitSize(stan_fit)
```

Arguments

`stan_fit`	A stanfit object.

Value

A stanfit object with a reduced size.

`rmdcev` *rmdcev: Estimating and simulating multiple discrete-continuous extreme value (MDCEV) demand models*

Description

The rmdcev R package estimates and simulates multiple discrete-continuous extreme value (MDCEV) demand models (also known as Kuhn-Tucker demand models) with observed and unobserved individual heterogneity (Bhat (2008) <doi.org/10.1016/j.trb.2007.06.002>). Fixed parameter, latent class, and random parameter models can be estimated. These models are estimated using maximum likelihood or Bayesian estimation techniques and are implemented in Stan, which is a C++ package for performing full Bayesian inference (see Stan Development Team (2018) <http://mc-stan.org>). The package also includes demand simulation (Pinjari and Bhat (2011) <https://repositories.lib.utexas.edu/handle/2152/23880>) and welfare simulation (Lloyd-Smith (2018) <doi.org/10.1016/j.jocm.2017.12.002>).

Author(s)

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

References

Bhat, CR (2008). The multiple discrete-continuous extreme value (MDCEV) model: Role of utility function parameters, identification considerations, and model extensions. Transportation Research Part B: Methodological, 42(3): 274-303.(link) Lloyd-Smith, P (2018). A new approach to calculating welfare measures in Kuhn-Tucker demand models. Journal of Choice Modeling 26:19–27. (link) Pinjari, AR, Bhat, CR (2011). Computationally Efficient Forecasting Procedures for Kuhn-Tucker Consumer Demand Model Systems: Application to Residential Energy Consumption Analysis. Department of Civil and Environmental Engineering, University of South Florida. (link) Stan Development Team (2018). RStan: the R interface to Stan. R package version 2.18.2. (link)

Index