Package ‘BLPestimatoR’

December 11, 2019

Type Package
Title Performs a BLP Demand Estimation
Version 0.3.2
Author Daniel Brunner (aut), Constantin Weiser (ctr), Andre Romahn (ctr)
Maintainer Daniel Brunner <daniel.brunner@hhu.de>
Description Provides the estimation algorithm to perform the demand estimation described in Berry, Levinsohn and Pakes (1995) <DOI:10.2307/2171802>. The routine uses analytic gradients and offers a large number of implemented integration methods and optimization routines.
License GPL-3
LazyData TRUE
Imports Rcpp (>= 0.11.0), mvQuad, numDeriv, randtoolbox, Formula, stats, Matrix
LinkingTo Rcpp, RcppArmadillo
RoxygenNote 6.1.1
NeedsCompilation yes
Suggests knitr, rmarkdown
VignetteBuilder knitr
Depends R (>= 3.5.0)
Repository CRAN
Date/Publication 2019-12-11 14:20:16 UTC

R topics documented:

  BLP_data ................................. 2
  demographicData_cereal .................. 4
  dstdDelta_wrap ............................ 5
  dstdtheta_wrap ............................ 6
  dummies_cars ............................. 7
  estimateBLP ............................... 8
### BLP_data

Prepares data and parameters related to the BLP algorithm for estimation.

#### Description

Prepares data and parameters related to the BLP algorithm for estimation.

#### Usage

```r
BLP_data(model, market_identifier, product_identifier, par_delta,
group_structure = NULL, additional_variables = NULL, productData,
demographic_draws, integration_accuracy, integration_method,
integration_draws, integration_weights, integration_seed,
blp_inner_tol = 1e-09, blp_inner_maxit = 10000)
```

#### Arguments

- **model**: the model to be estimated in R's formula syntax,
- **market_identifier**: character specifying the market identifier (variable name must be included in productData),
- **product_identifier**: character specifying the product identifier (variable name must be included in productData),
- **par_delta**: optional: numeric vector with values for the mean utility (variable name must be included in productData),
- **group_structure**: optional: character specifying a group structure for clustered standard errors (variable name must be included in productData),
- **additional_variables**: optional: character vector specifying variables you want to keep for later analysis (variable names must be included in productData)
BLP_data

productData data.frame with product characteristics,
demographic_draws optional: list with demographic draws for each market to consider observed heterogeneity (see details),
integration_accuracy integer specifying integration accuracy,
integration_method character specifying integration method,
integration_draws numeric matrix of manually provided integration draws (see details),
integration_weights numeric vector of manually provided integration weights,
integration_seed seed for the draws of Monte Carlo based integration,
blp_inner_tol tolerance for the contraction mapping (default: 1e-9),
blp_inner_maxit maximum iterations for the contraction mapping (default: 10000)

Details

For any form of user provided integration draws, i.e. integration_draws (unobserved heterogeneity) or demographic_draws (observed heterogeneity), list entries must be named and contain the variable market_identifier to allow market matching. Each line in these list entries contains the draws for one market. In case of unobserved heterogeneity, list names must match the random coefficients from the model formula. The par_delta argument provides the variable name for mean utilities. For example, in the estimation algorithm these values are used as starting guesses in the contraction mapping. Another example is the evaluation of the GMM, which is also based on the provided mean utilities. If you need to update par_delta or any other variable in the data object, use update_BLP_data.

Value

Returns an object of class blp_data.

Examples

K<-2 #number of random coefficients
data <- simulate_BLP_dataset(nmkt = 25, nbrn = 20,
Xlin = c("price", "x1", "x2", "x3", "x4", "x5"),
Xexo = c("x1", "x2", "x3", "x4", "x5"),
Xrandom = paste0("x",1:K),instruments = paste0("iv",1:10),
true.parameters = list(Xlin.true.except.price = rep(0.2,5),
Xlin.true.price = -0.2,
Xrandom.true = rep(2,K),
instrument.effects = rep(2,10),
instrument.Xexo.effects = rep(1,5)),
price.endogeneity = list(mean.xi = -2,
mean.eita = 0,
cov = cbind(c(1,0.7), c(0.7,1))),
demographicData_cereal

Draws for observed heterogeneity in Nevo’s cereal example.

Description

Draws for observed heterogeneity in Nevo’s cereal example.

Usage

demographicData_cereal

Format

Draws for observed heterogeneity for each demographic.

- **cdid**: market identifier,
- **draws**: 20 draws differing across markets.

Source


```r
printlevel = 0, seed = 234234 )

model <- as.formula("shares ~ price + x1 + x2 + x3 + x4 + x5 |
                   x1 + x2 + x3 + x4 + x5 |
                   0+ x1 + x2 |
                   iv1 + iv2 + iv3 + iv4 + iv5 + iv6 + iv7 + iv8 + iv9 + iv10")

blp_data <- BLP_data(model = model, market_identifier="cdid",
                      product_id = "prod_id",
                      productData = data,
                      integration_method = "MLHS",
                      integration_accuracy = 40,
                      integration_seed = 1)
```
dstddelta_wrap

Calculates derivatives of all shares with respect to all mean utilities in a given market.

Description

Calculates derivatives of all shares with respect to all mean utilities in a given market.

Usage

dstddelta_wrap(blp_data, par_theta2, market, printLevel = 1)

Arguments

blp_data        data object created by the function BLP_data,
par_theta2      matrix with column and rownames providing a starting value for the optimization routine (see details),
market          character specifying the market in which derivatives are calculated,
printLevel      level of output information (default = 1)

Details

NA's in par_theta2 entries indicate the exclusion from estimation, i.e. the coefficient is assumed to be zero. If only unobserved heterogeneity is used (no demographics), the column name of par_theta2 must be "unobs_sd". With demographics the colnames must match the names of provided demographics (as in demographic_draws) and "unobs_sd". Row names of par_theta2 must match random coefficients as specified in model. Constants must be named "(Intercept)".

Value

Returns a numeric matrix with derivatives. Cell in row i and col j is the derivative of share i with respect to mean utility j.

Examples

K<-2 #number of random coefficients
data <- simulate_BLP_dataset(nmkt = 25, nbrn = 20,
    Xlin = c("price", "x1", "x2", "x3", "x4", "x5"),
    Xexo = c("x1", "x2", "x3", "x4", "x5"),
    Xrandom = paste0("x",1:K),instruments = paste0("iv",1:10),
    true.parameters = list(Xlin.true.except.price = rep(0.2,5),
                       Xlin.true.price = -0.2,
                       Xrandom.true = rep(2,K),
                       instrument.effects = rep(2,10),
                       instrument.Xexo.effects = rep(1,5)),
    price.endogeneity = list( mean.xi = -2,
                       mean.eita = 0,
                       cov = cbind( c(1,0.7), c(0.7,1)))),
dstdtheta_wrap

Calculates derivatives of all shares with respect to all non-linear parameters in a given market.

Description
Calculates derivatives of all shares with respect to all non-linear parameters in a given market.

Usage

dstdtheta_wrap(blp_data, par_theta2, market, printLevel = 1)

Arguments

  blp_data        data object created by the function BLP_data,
  par_theta2      matrix with column and rownames providing a starting value for the optimization routine (see details),
  market          character specifying the market in which derivatives are calculated,
  printLevel      level of output information (default = 1)

Details
NA’s in par_theta2 entries indicate the exclusion from estimation, i.e. the coefficient is assumed to be zero. If only unobserved heterogeneity is used (no demographics), the column name of par_theta2 must be “unobs_sd”. With demographics the colnames must match the names of provided demographics (as in demographic_draws) and “unobs_sd”. Row names of par_theta2 must match random coefficients as specified in model. Constants must be named “(Intercept)”.

model <- as.formula("shares ~ price + x1 + x2 + x3 + x4 + x5 |
                     x1 + x2 + x3 + x4 + x5 |
                     x1 + x2 |
                     iv1 + iv2 + iv3 + iv4 + iv5 + iv6 + iv7 + iv8 + iv9 + iv10")

blp_data <- BLP_data(model = model, market_identifier="cdid",
                     product_id = "prod_id",
                     productData = data,
                     integration_method = "MLHS",
                     integration_accuracy = 40,
                     integration_seed = 1)

theta2 <- matrix(c(0.5, 2), nrow=2)
rownames(theta2) <- c("x1", "x2")
colnames(theta2) <- "unobs_sd"

derivatives2 <- dstddelta_wrap( blp_data=blp_data,
                               par_theta2 = theta2,
                               market = 2)
Value

Returns a numeric matrix with derivatives. Cell in row i and col j is the derivative of share i with respect to parameter j.

Examples

K<-2 #number of random coefficients
data <- simulate_BLP_dataset(nmkt = 25, nbrn = 20,
Xlin = c("price", "x1", "x2", "x3", "x4", "x5"),
Xexo = c("x1", "x2", "x3", "x4", "x5"),
Xrandom = paste("x",1:K),instruments = paste("iv",1:10),
true.parameters = list(Xlin.true.except.price = rep(0.2,5),
Xlin.true.price = -0.2,
Xrandom.true = rep(2,K),
instrument.effects = rep(2,10),
instrument.Xexo.effects = rep(1,5),
price.endogeneity = list( mean.xi = -2,
mean.eita = 0,
cov = cbind( c(1,0.7), c(0.7,1)) ),
printlevel = 0, seed = 234234 )

model <- as.formula("shares ~ price + x1 + x2 + x3 + x4 + x5 |
x1 + x2 + x3 + x4 + x5 |
0+ x1 + x2 |
iv1 + iv2 + iv3 + iv4 + iv5 + iv6 + iv7 + iv8 + iv9 + iv10")

blp_data <- BLP_data(model = model, market_identifier="cdid",
product_id = "prod_id",
productData = data,
integration_method = "MLHS",
integration_accuracy = 40,
integration_seed = 1)

theta2 <- matrix(c(0.5,2), nrow=2)
rownames(theta2) <- c("x1", "x2")
colnames(theta2) <- "unobs_sd"

derivatives1 <- dstdtheta_wrap( blp_data=blp_data,
par_theta2 = theta2,
market = 2)

---

dummies_cars

Ownership matrix in BLP’s car example.

Description

Ownership matrix in BLP’s car example.
Usage

dummies_cars

Format

Dummy variables.

column i 1, if product in row j is produced by firm i, 0 otherwise

Source


estimateBLP Performs a BLP demand estimation.

Description

Performs a BLP demand estimation.

Usage

estimateBLP(blp_data, par_theta2, solver_method = "BFGS", solver_maxit = 10000, solver_reltol = 1e-06, standardError = "heteroskedastic", extremumCheck = FALSE, printLevel = 2, ...)

Arguments

blp_data data object created by the function BLP_data,
par_theta2 matrix with column and rownames providing a starting value for the optimization routine (see details),
solver_method character specifying the solver method in optim (further arguments can be passed to optim by ...)
solver_maxit integer specifying maximum iterations for the optimization routine (default=10000),
solver_reltol integer specifying tolerance for the optimization routine (default= 1e-6),
standardError character specifying assumptions about the GMM residual (homoskedastic, heteroskedastic (default), or cluster)
extremumCheck if TRUE, second derivatives are checked for the existence of minimum at the point estimate (default = FALSE),
printLevel level of output information ranges from 0 (no GMM results) to 4 (every norm in the contraction mapping)
... additional arguments for optim
Details

NA's in \texttt{par\_theta2} entries indicate the exclusion from estimation, i.e. the coefficient is assumed to be zero. If only unobserved heterogeneity is used (no demographics), the column name of \texttt{par\_theta2} must be "unobs\_sd". With demographics the colnames must match the names of provided demographics (as in \texttt{demographic\_draws}) and "unobs\_sd". Row names of \texttt{par\_theta2} must match random coefficients as specified in \texttt{model}. Constants must be named "(Intercept)".

Value

Returns an object of class "blp\_est". This object contains, among others, all estimates for preference parameters and standard errors.

Examples

\begin{verbatim}
K<-2 #number of random coefficients
data <- simulate_BLP_dataset(nmkt = 25, nbrn = 20,
    Xlin = c("price", "x1", "x2", "x3", "x4", "x5"),
    Xexo = c("x1", "x2", "x3", "x4", "x5"),
    Xrandom = paste0("x",1:K),
    instruments = paste0("iv",1:10),
    true.parameters = list(Xlin.true.except.price = rep(0.2,5),
        Xlin.true.price = -0.2,
        Xrandom.true = rep(2,K),
        instrument.effects = rep(2,10),
        instrument.Xexo.effects = rep(1,5)),
    price.endogeneity = list(mean.xi = -2,
        mean.eita = 0,
        cov = cbind(c(1,0.7), c(0.7,1)))),
    printlevel = 0, seed = 234234 )

model <- as.formula("shares ~ price + x1 + x2 + x3 + x4 + x5 |
    x1 + x2 + x3 + x4 + x5 |
    0+ x1 + x2 |
    iv1 + iv2 + iv3 + iv4 + iv5 + iv6 + iv7 + iv8 + iv9 + iv10")

blp_data <- BLP_data(model = model, market_identifier="cdid",
    product_id = "prod_id",
    productData = data,
    integration_method = "MLHS",
    integration_accuracy = 40,
    integration_seed = 1)

theta_guesses <- matrix(c(0.5,2), nrow=2)
rownames(theta_guesses) <- c("x1","x2")
colnames(theta_guesses) <- "unobs\_sd"

blp_est <- estimateBLP(blp_data = blp_data,
    par\_theta2 = theta_guesses,
    extremumCheck = FALSE,
    printLevel = 1)

summary(blp_est)
\end{verbatim}
getDelta_wrap

Performs a contraction mapping for a given set of non-linear parameters.

Description

Performs a contraction mapping for a given set of non-linear parameters.

Usage

getDelta_wrap(blp_data, par_theta2, printLevel = 1)

Arguments

blp_data data object created by the function BLP_data,
par_theta2 matrix with column and rownames providing a starting value for the optimization routine (see details),
printLevel level of output information (default = 1)

Details

NA's in par_theta2 entries indicate the exclusion from estimation, i.e. the coefficient is assumed to be zero. If only unobserved heterogeneity is used (no demographics), the column name of par_theta2 must be "unobs_sd". With demographics the colnames must match the names of provided demographics (as in demographic_draws) and "unobs_sd". Row names of par_theta2 must match random coefficients as specified in model. Constants must be named "(Intercept)".

Starting guesses for the contraction mapping are provided with BLP_data.

Value

Returns an object of class "blp_cm" with results from the contraction mapping.

delta resulting vector of mean utilities after the contraction mapping
counter inner iterations needed to convergence
sij market share integral evaluations for each product (in rows) for the final mean utility

Examples

K<-2 #number of random coefficients
data <- simulate_BLP_dataset(nmkt = 25, nbrn = 20,
Xlin = c("price", "x1", "x2", "x3", "x4", "x5"),
Xexo = c("x1", "x2", "x3", "x4", "x5"),
Xrandom = paste0("x",1:K),instruments = paste0("iv",1:10),
true.parameters = list(Xlin.true.except.price = rep(0.2,5),
Xlin.true.price = -0.2,
Xrandom.true = rep(2,K),
instrument.effects = rep(2,10),
getJacobian_wrap

Calculating the Jacobian for a given set of non-linear parameters and mean utilities.

Description

Calculating the Jacobian for a given set of non-linear parameters and mean utilities.

Usage

getJacobian_wrap(blpm.data, par-theta2, printLevel = 1)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>blp.data</td>
<td>data object created by the function BLIP_data.</td>
</tr>
<tr>
<td>par-theta2</td>
<td>matrix with column and rownames providing the evaluation point (see details).</td>
</tr>
<tr>
<td>printLevel</td>
<td>level of output information (default = 1)</td>
</tr>
</tbody>
</table>
Details

NA’s in par_theta2 entries indicate the exclusion from estimation, i.e. the coefficient is assumed to be zero. If only unobserved heterogeneity is used (no demographics), the column name of par_theta2 must be "unobs_sd". With demographics the colnames must match the names of provided demographics (as in demographic_draws) and "unobs_sd". Row names of par_theta2 must match random coefficients as specified in model. Constants must be named "(Intercept)".

Value

Returns a matrix with the jacobian (products in rows, parameters in columns).

Examples

```r
K <- 2  # number of random coefficients
data <- simulate_BLP_dataset(nmkt = 25, nbrn = 20,
                            Xlin = c("price", "x1", "x2", "x3", "x4", "x5"),
                            Xexo = c("x1", "x2", "x3", "x4", "x5"),
                            Xrandom = paste0("x",1:K),
                            instruments = paste0("iv",1:10),
                            true.parameters = list(Xlin.true.except.price = rep(0.2,5),
                                                  Xlin.true.price = -0.2,
                                                  Xrandom.true = rep(2,K),
                                                  instrument.effects = rep(2,10),
                                                  instrument.Xexo.effects = rep(1,5)),
                            price.endogeneity = list(mean.xi = -2,
                                                      mean.eita = 0,
                                                      cov = cbind(c(1,0.7), c(0.7,1))),
                            printlevel = 0, seed = 234234)

model <- as.formula("shares ~ price + x1 + x2 + x3 + x4 + x5 |
                     x1 + x2 + x3 + x4 + x5 |
                     0+ x1 + x2 |
                     iv1 + iv2 + iv3 + iv4 + iv5 + iv6 + iv7 + iv8 + iv9 + iv10")

blp_data <- BLP_data(model = model, market_identifier="cdid",
                      product_id = "prod_id",
                      productData = data,
                      integration_method = "MLHS",
                      integration_accuracy = 40,
                      integration_seed = 1)

theta_guesses <- matrix(c(0.5,2), nrow=2)
rownames(theta_guesses) <- c("x1","x2")
colnames(theta_guesses) <- "unobs_sd"

jacobian <- getJacobian_wrap(blp_data=blp_data,
                               par_theta2 = theta_guesses,
                               printLevel = 2)

head(jacobian)
```
getShareInfo

Calculates information related to predicted shares for a given set of non-linear parameters and data.

Description
Calculates information related to predicted shares for a given set of non-linear parameters and data.

Usage
getShareInfo(blp_data, par_theta2, printLevel = 1)

Arguments
- `blp_data`: data object created by the function `BLP_data` (provides, among others, mean utilities and integration draws).
- `par_theta2`: matrix with column and rownames providing the evaluation point (see details).
- `printLevel`: level of output information (default = 1)

Value
Returns a list with information related to predicted shares.

Examples
```r
K <- 2  # number of random coefficients
data <- simulate_BLP_dataset(nmkt = 25, nbrn = 20,
  Xlin = c("price", "x1", "x2", "x3", "x4", "x5"),
  Xexo = c("x1", "x2", "x3", "x4", "x5"),
  Xrandom = paste0("x",1:K),
  instruments = paste0("iv",1:10),
  true.parameters = list(Xlin.true.except.price = rep(0.2,5),
    Xlin.true.price = -0.2,
    Xrandom.true = rep(2,K),
    instrument.effects = rep(2,10),
    instrument.Xexo.effects = rep(1,5)),
  price.endogeneity = list( mean.x1 = -2,
    mean.eita = 0,
    cov = cbind( c(1,0.7), c(0.7,1))),
  printlevel = 0, seed = 234234 )

model <- as.formula("shares ~ price + x1 + x2 + x3 + x4 + x5 |
  x1 + x2 + x3 + x4 + x5 |
  0+ x1 + x2 |
  iv1 + iv2 + iv3 + iv4 + iv5 + iv6 + iv7 + iv8 +iv9 +iv10")

blp_data <- BLP_data(model = model, market_identifier="cdid",
  product_id = "prod_id",
  productData = data,
  integration_method = "MLHS",
  integration_points = 5, integration_level = 3, integration_order = 3,)
```
get_elasticities <- matrix(c(0.5,2), nrow=2)
rownames(get_elasticities) <- c("x1","x2")
colnames(theta_guesses) <- "unobs_sd"

shares <- getShareInfo(blp_data=blp_data, 
par_theta2 = theta_guesses, 
printLevel = 4)
Examples

K<-2 #number of random coefficients
data <- simulate_BLP_dataset(nmkt = 25, nbrn = 20,
Xlin = c("price", "x1", "x2", "x3", "x4", "x5"),
Xexo = c("x1", "x2", "x3", "x4", "x5"),
Xrandom = paste0("x",1:K),instruments = paste0("iv",1:10),
true.parameters = list(Xlin.true.except.price = rep(0.2,5),
Xlin.true.price = -0.2,
Xrandom.true = rep(2,K),
instrument.effects = rep(2,10),
instrument.Xexo.effects = rep(1,5)),
price.endogeneity = list( mean.xi = -2,
mean.eita = 0,
cov = cbind( c(1,0.7), c(0.7,1)) ),
printlevel = 0, seed = 234234 )

model <- as.formula("shares ~ price + x1 + x2 + x3 + x4 + x5 |
x1 + x2 + x3 + x4 + x5 |
0+ x1 + x2 |
iv1 + iv2 + iv3 + iv4 + iv5 + iv6 + iv7 + iv8 +iv9 +iv10")

blp_data <- BLP_data(model = model, market_identifier="cdid",
product_id = "prod_id",
productData = data,
integration_method = "MLHS",
integration_accuracy = 40,
integration_seed = 1)

theta_guesses <- matrix(c(0.5,2), nrow=2)
rownames(theta_guesses) <- c("x1","x2")
colnames(theta_guesses) <- "unobs_sd"

shareObj <- getShareInfo( blp_data=blp_data,
par_theta2 = theta_guesses,
printLevel = 1)

gmm_obj_wrap
Calculating the GMM objective for a given set of non-linear parameters.
Description

Calculating the GMM objective for a given set of non-linear parameters.

Usage

gmm_obj_wrap(blp_data, par_theta2, printLevel = 2)

Arguments

- **blp_data**: data object created by the function `BLP_data`.
- **par_theta2**: matrix with column and rownames providing a starting value for the optimization routine (see details).
- **printLevel**: level of output information ranges from 1 (no GMM results) to 4 (every norm in the contraction mapping).

Details

NA's in `par_theta2` entries indicate the exclusion from estimation, i.e. the coefficient is assumed to be zero. If only unobserved heterogeneity is used (no demographics), the column name of `par_theta2` must be "unobs_sd". With demographics the colnames must match the names of provided demographics (as in `demographic_drawing`) and "unobs_sd". Row names of `par_theta2` must match random coefficients as specified in `model`. Constants must be named "(Intercept)".

Value

Returns a list with results from the GMM evaluation.

- **local_min**: GMM point evaluation
- **gradient**: GMM derivative with respect to non-linear parameters
- **delta**: result of the contraction mapping
- **xi**: residuals of GMM evaluation

Examples

```r
K <- 2  # number of random coefficients
data <- simulate_BLP_dataset(nmkt = 25, nbrn = 20,
    Xlin = c("price", "x1", "x2", "x3", "x4", "x5"),
    Xexo = c("x1", "x2", "x3", "x4", "x5"),
    Xrandom = paste0("x",1:K),
    instruments = paste0("iv",1:10),
    true.parameters = list(Xlin.true.except.price = rep(0.2,5),
        Xlin.true.price = -0.2,
        Xrandom.true = rep(2,K),
        instrument.effects = rep(2,10),
        instrument.Xexo.effects = rep(1,5)),
    price.endogeneity = list( mean.xi = -2,
        mean.eita = 0,
        cov = cbind( c(1,0.7), c(0.7,1) ) ),
    printlevel = 0, seed = 234234)
```
model <- as.formula("shares ~ price + x1 + x2 + x3 + x4 + x5 |
  x1 + x2 + x3 + x4 + x5 |
  0+ x1 + x2 |
  iv1 + iv2 + iv3 + iv4 + iv5 + iv6 + iv7 + iv8 + iv9 + iv10 ")

blp_data <- BLP_data(model = model, market_identifier="cdid",
  product_id = "prod_id",
  productData = data,
  integration_method = "MLHS",
  integration_accuracy = 40,
  integration_seed = 1)

theta_guesses <- matrix(c(0.5,2), nrow=2)
rownames(theta_guesses) <- c("x1","x2")
colnames(theta_guesses) <- "unobs_sd"

gmm <- gmm_obj_wrap( blp_data=blp_data,
  par_theta2 = theta_guesses,
  printLevel = 2)

gmm$local_min

originalDraws_cereal

Description

Draws for unobserved heterogeneity in Nevo’s cereal example.

Usage

originalDraws_cereal

Format

Each list entry contains draws (unobserved heterogeneity) for a random coefficient.

cdid  market identifier,
draws_ 20 draws differing across markets.

Source

Description

Product data of BLP’s car example.

Usage

productData_cars

Format

A data frame with product data of 2217 cars in 20 markets.

- **share**: car market share,
- **price**: car price,
- **hpwt**: horsepower-weight ratio,
- **air**: 1, if car has air conditioning, 0 otherwise,
- **mpg**: market identifier,
- **space**: length times width of the car,
- **const**: constant,
- **id**: uniquely identifies a car,
- **cdid**: uniquely identifies the market of a product,
- **firmid**: uniquely identifies the firm of a product (corresponds to column number in the ownership matrix).

Source


Description

Product data of Nevo’s cereal example.

Usage

productData_cereal
Format

A data frame with product data of 24 cereals in each of 94 markets.

share cereals market share,
price cereals price,
const constant,
sugar cereals sugar,
mushy cereals mushy,
cdid market identifier,
product_id uniquely identifies a product in a market,
productdummy uniquely identifies a product in a market,
IV1 1. instrument,
IV2 2. instrument,
IV3 3. instrument,
IV4 4. instrument,
IV5 5. instrument,
IV6 6. instrument,
IV7 7. instrument,
IV8 8. instrument,
IV9 9. instrument,
IV10 10. instrument,
IV11 11. instrument,
IV12 12. instrument,
IV13 13. instrument,
IV14 14. instrument,
IV15 15. instrument,
IV16 16. instrument,
IV17 17. instrument,
IV18 18. instrument,
IV19 19. instrument,
IV20 20. instrument

Source

**simulate_BLP_dataset**  
*This function creates a simulated BLP dataset.*

**Description**

This function creates a simulated BLP dataset.

**Usage**

```r
simulate_BLP_dataset(nmkt, nbrn, Xlin, Xexo, Xrandom, instruments, 
true.parameters = list(), price.endogeneity = list(mean.xi = -2, 
mean.eita = 0, cov = cbind(c(1, 0.7), c(0.7, 1))), printlevel = 1, 
seed)
```

**Arguments**

- `nmkt`: number of markets
- `nbrn`: number of products
- `Xlin`: character vector specifying the set of linear variables
- `Xexo`: character vector specifying the set of exogenous variables (subset of `Xlin`)
- `Xrandom`: character vector specifying the set of random coefficients (subset of `Xlin`)
- `instruments`: character vector specifying the set of instrumental variables
- `true.parameters`: list with parameters of the DGP
  - `Xlin.true.except.price`: "true" linear coefficients in utility function except price
  - `Xlin.true.price`: "true" linear price coefficient in utility function
  - `Xrandom.true`: "true" set of random coefficients
  - `instrument.effects`: "true" coefficients of instrumental variables to explain endogenous price
  - `instrument.Xexo.effects`: "true" coefficients of exogenous variables to explain endogenous price
- `price.endogeneity`: list with arguments of the multivariate normal distribution
  - `mean.xi`: controls for the mean of the error term in the utility function
  - `mean.eita`: controls for the mean of the error term in the price function
  - `cov`: controls for the covariance of `xi` and `eita`
- `printlevel`: 0 (no output) 1 (summary of generated data)
- `seed`: seed for the random number generator
Details

The dataset is balanced, so every market has the same amount of products. Only unobserved heterogeneity can be considered. Variables that enter the equation as a Random Coefficient or exogenously must be included in the set of linear variables. The parameter list argument specifies the "true" effect on the individual utility for each component. Prices are generated endogenous as a function of exogenous variables and instruments, where the respective effect sizes are specified in instrument.effects and instrument.Xexo.effects. Error terms \( x_i \) and \( eita \) are drawn from a multivariate normal distribution, whose parameters can be set in price.endogeneity. Market shares are generated by MLHS integration rule with 10000 nodes.

Value

Returns a simulated BLP dataset.

Examples

\[ K<-2 \] # number of random coefficients

---

| theta_guesses_cereal | Parameter starting guesses for Nevo's cereal example. |

Description

Parameter starting guesses for Nevo’s cereal example.

Usage

theta_guesses_cereal

Format

A matrix with 4 random coefficients (rows) and columns for 4 demographics and one unobserved heterogeneity column (5 cols in total).

Source

update_BLP_data  
*Updates the set of linear, exogenous, random coefficient, share or mean utility variable in the data object.*

**Description**

Updates the set of linear, exogenous, random coefficient, share or mean utility variable in the data object.

**Usage**

```
update_BLP_data(data_update, blp_data)
```

**Arguments**

- `data_update`  
data.frame with variables to update (must contain the market_identifier and product_identifier variables as in `blp_data`).
- `blp_data`  
data object created by the function `BLP_data`.

**Value**

Returns an object of class `blp_data`.

**Examples**

```r
K <- 2  # number of random coefficients
data <- simulate_BLP_dataset(nmkt = 25, nbrn = 20,  
  Xlin = c("price", "x1", "x2", "x3", "x4", "x5"),  
  Xexo = c("x1", "x2", "x3", "x4", "x5"),  
  Xrandom = paste0("x", 1:K),  
  instruments = paste0("iv", 1:10),  
  true.parameters = list(Xlin.true.except.price = rep(0.2, 5),  
    Xlin.true.price = -0.2,  
    Xrandom.true = rep(2, K),  
    instrument.effects = rep(2, 10),  
    instrument.Xexo.effects = rep(1, 5)),  
  price.endogeneity = list(mean.xi = -2,  
    mean.eita = 0,  
    cov = cbind(c(1, 0.7), c(0.7, 1))),  
  printlevel = 0, seed = 234234)

model <- as.formula("shares ~ price + x1 + x2 + x3 + x4 + x5 |  
  x1 + x2 + x3 + x4 + x5 |  
  0+ x1 + x2 |  
  iv1 + iv2 + iv3 + iv4 + iv5 + iv6 + iv7 + iv8 + iv9 + iv10")

blp_data <- BLP_data(model = model, market_identifier="cdid",  
  product_id = "prod_id",  
  productData = data,
```
integration_method = "MLHS",
integration_accuracy = 40,
integration_seed = 1)

new_data <- data.frame(price = seq(1,10,length.out=500),
x1 = seq(2,10,length.out=500),
ccd = sort(rep(1:25,20)),
prod_id = rep(1:20,25) )
blp_data_example_updated <- update_BLP_data(blp_data = blp_data,
data_update = new_data)

---

**w_guesses_cereal** Mean utility starting guesses for Nevo’s cereal example.

**Description**
Mean utility starting guesses for Nevo’s cereal example.

**Usage**

w_guesses_cereal

**Format**
A numeric vector of 2256 values.

**Source**
## Index

*Topic **datasets**
  - `demographicData_cereal`, 4
  - `dummies_cars`, 7
  - `originalDraws_cereal`, 17
  - `productData_cars`, 18
  - `productData_cereal`, 18
  - `theta_guesses_cereal`, 21
  - `w_guesses_cereal`, 23

BLP data, 2
  - `demographicData_cereal`, 4
  - `dstddelta_wrap`, 5
  - `dstdtheta_wrap`, 6
  - `dummies_cars`, 7
  - `estimateBLP`, 8
  - `get_elasticities`, 14
  - `getDelta_wrap`, 10
  - `getJacobian_wrap`, 11
  - `getShareInfo`, 13
  - `gmm_obj_wrap`, 15
  - `originalDraws_cereal`, 17
  - `productData_cars`, 18
  - `productData_cereal`, 18
  - `simulate_BLP_dataset`, 20
  - `theta_guesses_cereal`, 21
  - `update_BLP_data`, 22
  - `w_guesses_cereal`, 23