Package ‘idefix’

January 17, 2018

Type Package

Title Efficient Designs for Discrete Choice Experiments

Version 0.2.4

Author Frits Traets [aut, cre]

Maintainer Frits Traets <frits.traets@kuleuven.be>

Description Generates efficient designs for discrete choice experiments based on the multinomial logit model, and individually adapted designs for the mixed multinomial logit model. The generated designs can be presented on screen and choice data can be gathered using a shiny application. Crabbe M, Akinc D and Vandebroek M (2014) <doi:10.1016/j.trb.2013.11.008>.

License GPL-3

Depends R (>= 3.1.1), shiny

LazyData TRUE

ByteCompile TRUE

Imports dplyr, gtools, MASS, maxLik, Rdpack, stats, scales, utils

RoxygenNote 6.0.1

RdMacros Rdpack

URL https://github.com/traets/idefix

NeedsCompilation no

Repository CRAN

Date/Publication 2018-01-17 12:40:54 UTC

R topics documented:

aggregate_design 2
Charbin 2
Datatrans 3
Decode 4
example_design 6
ImpsampMNL 6

LoadData 8
Modfed 8
Profiles 10
RespondMNL 11
SeqDB 12
SeqKL 13
SurveyApp 15

Index **19**

`aggregate_design` *Discrete choice aggregate design.*

Description

The dataset contains fictional data for seven participants, who each responded to eight choice sets with two alternatives. Each alternatives consists of three attributes who each contain three levels and are dummy coded.

Usage

```
data(aggregate_design)
```

Format

A matrix with 112 rows and 9 variables

`Charbin` *Character vector to binary vector.*

Description

Transforms a character vector with responses into a binary vector. Each alternative in each choice set wil be either 0 or 1. If the alternative was not chosen 0, and 1 if it was. The function can be used for example in a shiny application to transform the response vector received from `radioButtons` into a numeric vector that can be used for estimation.

Usage

```
Charbin(resp, alts, n.alts, no.choice = FALSE)
```

Arguments

`resp`	String vector containing input responses
`alts`	String vector containing all possible alternatives. The order should be the same as the order of the design matrix.
`n.alts`	The number of alternatives per choice set.
`no.choice`	Logical value indicating whether a no.choice option is provided or not. The default = `FALSE`.

Details

The `n.alts` denotes the number of alternatives a respondent could choose from, without counting a possible no choice option.

If `no.choice` is `TRUE` the first alternative specified in `alts` will be treated as a no choice option. If the no choice option was chosen all alternatives are zero for that choice set.

Value

A binary response vector with length equal to `length(resp) * length(n.alts)`.

Examples

```
# Observed Responses
resp <- c("alt1", "alt3", "alt2", "no.choice", "alt1")
# All possible alternatives
alts <- c("no.choice", "alt1", "alt2", "alt3")
# 3 alternatives + no.choice
Charbin(resp = resp, alts = alts, n.alts = 3, no.choice = TRUE)
```

`Datatrans` *Data transformation.*

Description

Transforms the data into the neccesary format in order to use estimation functions from different packages.

Usage

```
Datatrans(pkg, des, y, n.alts, n.sets, n.resp, n.par, no.choice, bin)
```

Arguments

pkg	Indicates the required package for estimation (`1 = rhierMnlRwMixture`, `2 = choicemodelr`, `3 = doHB` and `4 = rbprobitGibbs`).
des	A design matrix in which each row is a profile.
y	A numeric matrix. Each columnvector is the sequence of choices of a unique respondent. There can be `n.sets` rows with discrete values indicating the chosen alternative of that set, or there can be `n.sets * n.alts` rows with binary values indicating for each alternative whether it was chosen or not. In the latter case the `bin` argument should be `TRUE`.
n.alts	Numeric value indicating the number of alternatives per choice set.
n.sets	Numeric value indicating the number of choice sets.
n.resp	Numeric value indicating the number of respondents.
n.par	Numeric value indicating the number of model parameters that needs to be estimated.
no.choice	Logical value indicating whether a no choice response could be observed. This would be a `0` for each alternative.
bin	Logical value indicating whether the reponse matrix contains binary data (`TRUE`) or discrete data (`FALSE`). See `y`.

Value

The data ready to be used by the specified package.

Examples

```
# 3 Attributes, 2 are dummy coded and 1 continuous.
cs <- Profiles(lvls = c(2, 3, 2), coding = c("D", "C", "D"), c.lvls = list(c(2,4,6)))
p <- c(0.8, 0.2, -0.3) # parameter vector
# Generate design
des <- Modfed(cand.set = cs, n.sets = 8, n.alts = 2, alt.cte = c(0,0), par.draws = p)$des
# Generate responses
y <- RespondMNL(par = p, des = des, n.alts = 2)
y <- matrix(y, 16)
#  data
Datatrans(pkg = 4, des = des, y = y, n.alts = 2, n.sets = 8, n.resp = 1,
          n.par = 3, no.choice = FALSE, bin = TRUE)
```

`Decode` *Coded choice set to character choice set.*

Description

Transforms a coded choice set into a choice set containing character attribute levels, ready to be used in a survey.

Usage

```
Decode(set, lvl.names, coding, alt.cte = NULL, c.lvls = NULL)
```

Arguments

`set`	A numeric matrix which represents a choice set. Each row is a profile.
`lvl.names`	A list containing character vectors with the values of each level of each attribute.
`coding`	A character vector denoting the type of coding used for each attribute. See also `Profiles`.
`alt.cte`	A binary vector indicating for each alternative if an alternative specific constant is present. The default is `NULL`.
`c.lvls`	A list containing numeric vectors with the attributelevels for each continuous attribute. The default is `NULL`.

Details

In `lvl.names`, the number of character vectors in the list should equal the number of attributes in de choice set. The number of elements in each character vector should equal the number of levels for that attribute.

Valid arguments for `coding` are `C`, `D` and `E`. When using `C` the attribute will be treated as continuous and no coding will be applied. All possible levels should then be specified in `c.lvls`. If `D` (dummy coding) is used `contr.treatment` will be applied to that attribute. The first attribute wil be used as reference level. For `E` (effect coding) `contr.sum` is applied, in this case the last attributelevel is used as reference level.

Value

A character matrix which represents the choice set.

Examples

```
# Example without continuous attributes.
l <- c(3, 4, 2) # 3 Attributes.
c <- c("D", "E", "D") # Coding.
# All profiles.
p <- Profiles(lvls = l, coding = c)
cs <- p[c(4, 8), ] # Choice set
# Levels as they should appear in survey.
al <- list(
 c("$50", "$75", "$100"), # Levels attribute 1.
 c("2 min", "15 min", "30 min", "50 min"), # Levels attribute 2.
 c("bad", "good") # Levels attribute 3.
)
# Decode
Decode(set = cs, lvl.names = al, coding = c, alt.cte = c(0, 0))

# Example with continuous attribute.
l <- c(3, 4, 2) # 3 Attributes.
```

```
c <- c("D", "C", "D") # Coding.
cl <- list(c(50, 75, 80, 100))
# All profiles.
p <- Profiles(lvls = l, coding = c, c.lvls = cl)
cs <- p[c(4, 8), ] # Set.
a <- c(1, 0) # Alternative specific constant.
cs <- cbind(a, cs) # set with alt.cte
# Levels as they should appear in survey.
al <- list(
  c("$50", "$75", "$100"), # Levels attribute 1.
  c("50 min", "75 min", "80 min", "100 min"), # Levels attribute 2.
  c("bad", "good") # Levels attribute 3.
)
# Decode
Decode(set = cs, lvl.names = al, coding = c, alt.cte = c(1, 0), c.lvls = cl)
```

example_design *Discrete choice design.*

Description

This discrete choice design is generated using the `Modfed` function. There are 8 choice sets, each containig 2 alternatives (rows). The alternatives consist of 3 attributes (time, price, comfort) with each 3 levels, all of which are dummy coded (columns).

Usage

```
data(example_design)
```

Format

A matrix with 16 rows and 6 variables

ImpsampMNL *Importance sampling MNL*

Description

This function samples from the posterior distribution using importance sampling, assuming a multivariate normal prior distribution and a MNL likelihood.

Usage

```
ImpsampMNL(prior.mean, prior.covar, des, n.alts, y, m, b = 2)
```

Arguments

`prior.mean`	Numeric vector indicating the mean of the multivariate normal distribution (prior).
`prior.covar`	Covariance matrix of the prior distribution.
`des`	A design matrix in which each row is a profile. Can be generated with `Modfed`
`n.alts`	Numeric value indicating the number of alternatives per choice set.
`y`	A binary response vector. `RespondMNL` can be used to simulate respons data.
`m`	Numeric value. Number of draws = `base^m`.
`b`	Numeric value indicating the base. The default = 2.

Value

`sample`	Numeric vector with the (unweigthted) draws from the posterior distribution.
`weights`	Numeric vector with the associated weights of the draws.
`max`	Numeric vector with the estimated mode of the posterior distribution.
`covar`	Matrix representing the estimated variance covariance matrix.

Examples

```
# Importance sampling MNL
pm <- c(0.8, 0.3, 0.2, -0.3, -0.2) # Prior mean (4 parameters).
pc <- diag(length(pm)) # Prior variance
cs <- Profiles(lvls = c(3, 3), coding = c("E", "E"))
ps <- MASS::mvrnorm(n = 10, mu = pm, Sigma = pc) # 10 draws.
# Efficient design.
design <- Modfed(cand.set = cs, n.sets = 8, n.alts = 2, alt.cte = c(1,0), par.draws = ps)$design
# Respons.
resp <- RespondMNL(par = c(0.7, 0.6, 0.5, -0.5, -0.7), des = design, n.alts = 2)
# Parameters draws from posterior.
ImpsampMNL(prior.mean =  pm, prior.covar = pc, des = design, n.alts = 2, y = resp, m = 6)

# Importance sampling MNL
pm <- c(0.3, 0.2, -0.3, -0.2) # Prior mean (4 parameters).
pc <- diag(length(pm)) # Prior variance
cs <- Profiles(lvls = c(3, 3, 2), coding = c("D", "C", "D"), c.lvls = list(c(2,4,6)))
ac <- c(0, 0) # No alternative specific constants.
ps <- MASS::mvrnorm(n = 10, mu = pm, Sigma = pc) # 10 draws.
# Efficient design.
design <- Modfed(cand.set = cs, n.sets = 8, n.alts = 2, alt.cte = c(0,0), par.draws = ps)$design
# Respons
resp <- RespondMNL(par = c(0.6, 0.5, -0.5, -0.7), des = design, n.alts = 2)
# Parameters draws from posterior.
ImpsampMNL(prior.mean =  pm, prior.covar = pc, des = design, n.alts = 2, y = resp, m = 6)
```

LoadData *Load numeric choice data from directory*

Description

Reads all individual choice data files from a directory and concatenates those files into a single data file. Files containing either "num" or "char" will be read, with num indicating numeric data and char indicating character data. for more information see output of `SurveyApp`.

Usage

```
LoadData(data.dir, type)
```

Arguments

`data.dir`	A character string containing the directory to read from.
`type`	Character vector containing either num or char.

Value

A data frame containg the full design and all the responses of the combined data files that where found. Different files are indicated by an ID variable.

Modfed *Modified Federov algorithm for MNL models.*

Description

The algorithm swaps every profile of an initial design with candidate profiles. By doing this it tries to minimize the D(B)-error, based on a multinomial logit model.

Usage

```
Modfed(cand.set, n.sets, n.alts, alt.cte, par.draws, start.des = NULL,
  max.iter = Inf)
```

Arguments

`cand.set`	A numeric matrix in which each row is a possible profile. The `Profiles` function can be used to generate this.
`n.sets`	Numeric value indicating the number of choice sets.
`n.alts`	Numeric value indicating the number of alternatives per choice set.
`alt.cte`	A binary vector indicating for each alternative if an alternative specific constant is desired.

`par.draws`	A numeric vector containing the parameter values, or a numeric matrix in which each row is a draw from the multivariate prior parameter distribution.
`start.des`	A matrix in which each row is a profile. The number of rows equals `n.sets * n.alts`, and the number of columns equals the number of columns of `cand.set`. If not specified a random start design will be generated.
`max.iter`	A numeric value indicating the maximum number allowed iterations.

Details

Each iteration will loop through all profiles from the initial design, evaluating the change in D(B)-error for every profile from `cand.set`. The algorithm stops when an iteration occured without replacing a profile or when `max.iter` is reached.

By specifying a numeric vector in `par.draws`, the D-error will be calculated and the design will be optimised locally. By specifying a matrix, in which each row is a draw from a multivariate distribution, the DB-error will be calculated, and the design will be optimised globally. The number of columns should equal the number of parameter in `alt.cte` + the number of parameters in `cand.set`. This is also the order in which they should be sorted (first `alt.cte` parameters).

The DB-error is calculated by taking the mean over D-errors. It could be that for some draws the design results in an infinite D-error. The percentage of draws for which this was true for the final design can be found in the output `inf.error`.

Alternative specific constants can be specified in `alt.cte`. The lenght of this binary vector should equal `n.alts`, were `0` indicates the absence of an alternative specific constant and `1` the opposite.

Value

`design`	A numeric matrix wich contains an efficient design.
`error`	Numeric value indicating the D(B)-error of the design.
`inf.error`	Numeric value indicating the percentage of draws for which the D-error was `Inf`.
`prob.diff`	Numeric value indicating the difference between the alternative with the highest and the one with the lowest probability for each choice set. If a sample matrix was provided this is based on the average over all draws.

References

Cook RD and Nachtsheim CJ (1980). “A Comparison of Algorithms for Constructing Exact D-Optimal Designs.” *Technometrics*, **22**(3), pp. 315-324. ISSN 00401706, `http://www.jstor.org/stable/1268315`.

Examples

```
# D-efficient design
# 3 Attributes, 2 are dummy coded and 1 continuous (= 3 parameters).
cs <- Profiles(lvls = c(2, 3, 2), coding = c("D", "C", "D"), c.lvls = list(c(2, 4, 6)))
ps <- c(0.8, 0.2, -0.3) # Prior parameter vector
Modfed(cand.set = cs, n.sets = 8, n.alts = 2, alt.cte = c(0, 0), par.draws = ps)
```

```
# DB-efficient design.
# 3 Attributes with 2, 3 and 2 levels, all effect coded (= 4 parameters).
cs <- Profiles(lvls = c(2, 3, 2), coding = c("E", "E", "E"))
m <- c(0.8, 0.2, -0.3, -0.2, 0.7) # Prior mean (total = 5 parameters).
v <- diag(length(m)) # Prior variance.
set.seed(123)
ps <- MASS::mvrnorm(n = 10, mu = m, Sigma = v) # 10 draws.
Modfed(cand.set = cs, n.sets = 8, n.alts = 2, alt.cte = c(1, 0), par.draws = ps)
```

Profiles — *Profiles generation.*

Description

Function to generate all possible combinations of attribute levels (i.e. all possible profiles).

Usage

```
Profiles(lvls, coding, c.lvls = NULL)
```

Arguments

`lvls`	A numeric vector which contains for each attribute, the number of levels.
`coding`	Type op coding that needs to be used for each attribute.
`c.lvls`	A list containing numeric vectors with the attributelevels for each continuous attribute. The default is `NULL`.

Details

Valid arguments for `coding` are `C`, `D` and `E`. When using `C` the attribute will be treated as continuous and no coding will be applied. All possible levels should then be specified in `c.lvls`. If `D` (dummy coding) is used `contr.treatment` will be applied to that attribute. For `E` (effect coding) `contr.sum` will be applied.

Value

A numeric matrix which contains all possible profiles.

Examples

```
# Without continuous attributes
at.lvls <- c(3,4,2) # 3 Attributes with respectively 3, 4 and 2 levels.
c.type <- rep("E", length(at.lvls)) # All Effect coded.
Profiles(lvls = at.lvls, coding = c.type) # Generate profiles.

# With continuous attributes
at.lvls <- c(3,4,2) # 3 attributes with respectively 3, 4 and 2 levels.
# First attribute is dummy coded, second and third are continuous.
```

```
c.type <- c("D", "C", "C")
# Levels for continuous attributes, in the same order.
con.lvls <- list(c(4,6,8,10), c(7,9))
Profiles(lvls = at.lvls, coding = c.type, c.lvls = con.lvls)
```

RespondMNL *Response generation*

Description

Function to generate responses given parameter values and a design matrix, assuming a MNL model.

Usage

```
RespondMNL(par, des, n.alts, bin = TRUE)
```

Arguments

`par`	Numeric vector containing parameter values.
`des`	A design matrix in which each row is a profile. Can be generated with `Modfed`
`n.alts`	Numeric value indicating the number of alternatives per choice set.
`bin`	Indicates whether the returned value should be a binary vector or a discrete value which denotes the chosen alternative.

Value

Numeric vector indicating the chosen alternatives.

Examples

```
# 3 Attributes, 2 are dummy coded and 1 continuous.
cs <- Profiles(lvls = c(2, 3, 2), coding = c("D", "C", "D"), c.lvls = list(c(2,4,6)))
p <- c(0.8, 0.2, -0.3) # parameter vector
# Generate design
des <- Modfed(cand.set = cs, n.sets = 8, n.alts = 2, alt.cte = c(0,0), par.draws = p)$des
# Generate responses
y <- RespondMNL(par = p, des = des, n.alts = 2)
```

SeqDB *Sequential modified federov algorithm for MNL model.*

Description

Selects the choice set that minimizes the DB-error when added to an initial design, given (updated) parameter values.

Usage

```
SeqDB(des, cand.set, n.alts, par.draws, prior.covar, reduce = TRUE,
  weights = NULL)
```

Arguments

`des`	A design matrix in which each row is a profile. Can be generated with `Modfed`
`cand.set`	A numeric matrix in which each row is a possible profile. The `Profiles` function can be used to generate this.
`n.alts`	Numeric value indicating the number of alternatives per choice set.
`par.draws`	A matrix in which each row is a sample from the multivariate parameter distribution. See also `ImpsampMNL`.
`prior.covar`	Covariance matrix of the prior distribution.
`reduce`	Logical value indicating whether the candidate set should be reduced or not.
`weights`	A vector containing the weights of the draws. Default is `NULL`, See also `ImpsampMNL`.

Details

This algorithm is ideally used in an adaptive context. The algorithm will select the next DB-efficient choice set given parameter values and an initial design. In an adaptive context these parameter values are updated after each observed response.

The initial design `des` can be generated with `Modfed`. If alternative specific constants are included in the initial design, the algorithm will use the same for selecting the new choice set. Columns of `des` which contain ".cte" in their name are recognized as alternative specific columns.

The list of potential choice sets are created using `combinations`. If `reduce` is `TRUE`, `repeats.allowed = FALSE` and vice versa. If no alternative constants are used `reduce` should always be `TRUE`. When alternative specific constants are used `reduce` can be `TRUE` so that the algorithm will be faster, but the combinations of constants and profiles will not be evaluated exhaustively.

The `weights` can be used when the `par.draws` have weights. This is for example the case when parameter values are updated using `ImpsampMNL`.

Value

`set`	A matrix representing a DB efficient choice set.
`db.error`	A numeric value indicating the DB-error of the whole design.

References

Yu J, Goos P and Vandebroek M (2011). “Individually adapted sequential Bayesian conjoint-choice designs in the presence of consumer heterogeneity.” http://www.sciencedirect.com/science/article/pii/S0167811611000668.

Examples

```
# DB efficient choice set, given a design and parameter draws.
# Candidate profiles
cs <- Profiles(lvls = c(3, 3), coding = c("E", "E"))
m <- c(0.3, 0.2, -0.3, -0.2) # Prior mean (total = 5 parameters).
pc <- diag(length(m)) # Prior variance
set.seed(123)
ps <- MASS::mvrnorm(n = 10, mu = m, Sigma = pc) # 10 draws.
ac <- c(0, 0) # No alternative specific constants.
# Initial design.
des <- Modfed(cand.set = cs, n.sets = 6, n.alts = 2, alt.cte = ac, par.draws = ps)$design
# Efficient choice set to add.
SeqDB(des = des, cand.set = cs, n.alts = 2, par.draws = ps, prior.covar = pc)

# DB efficient choice set, given a design and parameter draws.
# Candidate profiles
cs <- Profiles(lvls = c(3, 3), coding = c("C", "E"), c.lvls = list(c(5,3,1)))
m <- c(0.7, 0.3, -0.3, -0.2) # Prior mean (4 parameters).
pc <- diag(length(m)) # Prior variance
set.seed(123)
ps <- MASS::mvrnorm(n = 10, mu = m, Sigma = pc) # 10 draws.
ac <- c(1, 0) # Alternative specific constant.
# Initial design.
des <- Modfed(cand.set = cs, n.sets = 6, n.alts = 2, alt.cte = ac, par.draws = ps)$design
# Efficient choice set to add.
SeqDB(des = des, cand.set = cs, n.alts = 2, par.draws = ps, prior.covar = pc)
```

SeqKL *Sequential Kullback-Leibler based algorithm for the MNL model.*

Description

Selects the choice set that maximizes the Kullback-Leibler divergence between prior parameter values and the expected posterior, assuming an MNL model.

Usage

```
SeqKL(cand.set, n.alts, alt.cte, par.draws, weights, reduce = TRUE)
```

Arguments

cand.set	A numeric matrix in which each row is a possible profile. The Profiles function can be used to generate this.
n.alts	Numeric value indicating the number of alternatives per choice set.
alt.cte	A binary vector indicating for each alternative if an alternative specific constant is desired.
par.draws	A matrix in which each row is a sample from the multivariate parameter distribution. See also ImpsampMNL.
weights	A vector containing the weights of the draws. Default is NULL, See also ImpsampMNL.
reduce	Logical value indicating whether the candidate set should be reduced or not.

Details

The algorithm selects the choice set that maximizes the Kullback-Leibler divergence between prior and expected posterior. Otherwisely framed the algorithm selects the choice set that maximizes the expected information gain.

Value

set	Numeric matrix containing the choice set that maximizes the expected KL divergence.
kl	Numeric value which is the Kullback leibler divergence.

References

Crabbe M, Akinc D and Vandebroek M (2014). "Fast algorithms to generate individualized designs for the mixed logit choice model." http://www.sciencedirect.com/science/article/pii/S0191261513002178.

Examples

```
# KL efficient choice set, given parameter draws.
# Candidate profiles
cs <- Profiles(lvls = c(3, 3), coding = c("E", "E"))
m <- c(0.3, 0.2, -0.3, -0.2) # Prior mean (4 parameters).
pc <- diag(length(m)) # Prior variance
set.seed(123)
ps <- MASS::mvrnorm(n = 10, mu = m, Sigma = pc) # 10 draws.
ac <- c(0, 0) # No alternative specific constants.
# Efficient choice set to add.
SeqKL(cand.set = cs, n.alts = 2, alt.cte = ac, par.draws = ps, weights = NULL)

# KL efficient choice set, given parameter draws.
# Candidate profiles
cs <- Profiles(lvls = c(3, 3), coding = c("C", "E"), c.lvls = list(c(5,3,1)))
m <- c(0.7, 0.3, -0.3, -0.2) # Prior mean (4 parameters).
pc <- diag(length(m)) # Prior variance
set.seed(123)
```

```
ps <- MASS::mvrnorm(n = 10, mu = m, Sigma = pc) # 10 draws.
ac <- c(1, 0) # Alternative specific constant.
# Efficient choice set to add.
SeqKL(cand.set = cs, n.alts = 2, alt.cte = ac, par.draws = ps, weights = NULL)
```

`SurveyApp` *Shiny application to generate a discrete choice survey.*

Description

This function starts a shiny application which puts choice sets on screen and saves the responses. The complete choice design can be provided in advance, or can be generated sequentially adaptively, or can be a combination of both.

Usage

```
SurveyApp(des = NULL, n.total, alts, atts, lvl.names, coding, buttons.text,
  intro.text, end.text, data.dir = NULL, c.lvls = NULL, crit = NULL,
  alt.cte = NULL, prior.mean = NULL, prior.covar = NULL,
  cand.set = NULL, m = NULL)
```

Arguments

`des`	A design matrix in which each row is a profile. Can be generated with `Modfed`
`n.total`	A numeric value indicating the total number of choice sets.
`alts`	A character vector containing the names of the alternatives.
`atts`	A character vector containing the names of the attributes.
`lvl.names`	A list containing character vectors with the values of each level of each attribute.
`coding`	A character vector denoting the type of coding used for each attribute. See also `Profiles`.
`buttons.text`	A string containing the text presented together with the option buttons.
`intro.text`	A string containing the text presented before the choice survey.
`end.text`	A string containing the text presented after the choice survey.
`data.dir`	A character string with the directory denoting where the data needs to be written. The default is NULL
`c.lvls`	A list containing numeric vectors with the attributelevels for each continuous attribute. The default is `NULL`.
`crit`	A string containing eihter KL or DB indicating the adaptive criterion to be used.
`alt.cte`	A binary vector indicating for each alternative if an alternative specific constant is present. The default is `NULL`.
`prior.mean`	Numeric vector indicating the mean of the multivariate normal distribution (prior).
`prior.covar`	Covariance matrix of the prior distribution.
`cand.set`	A numeric matrix in which each row is a possible profile. The `Profiles` function can be used to generate this.
`m`	Numeric value. Number of draws = `base^m`.

Details

A pregenerated design can be specified in `des`. This should be a matrix in which each row is a profile. This can be generated with `Modfed`, but is not necesarry.

If `n.total` = `nrow(des)` / `length(alts)`, the specified design will be put on screen, one set after the other, and the responses will be saved. If `n.total` > (`nrow(des)` / `length(alts)`), first the specified design will be shown and afterwards the remaining sets will be generated adaptively. If `des` = `NULL`, `n.total` sets will be generated adaptively.

Whenever adaptive sets will be generated, `crit`, `prior.mean`, `prior.covar`, `cand.set` and `m`, should be specified.

The names specified in `alts` will be used to label the choice alternatives. The names specified in `atts` will be used to name the attributes in the choice sets. The values of `lvl.names` will be used to create the values in the choice sets. See `Decode` for more details. The number of draws sampeled from the posterior preference distribution in the importance sampling algorithm used for adaptive sets can be specified with `m`, where the number is 2^`m`.

The text specified in `buttons.text` will be displayed above the buttons to indicate the preferred choice (for example: "indicate your preferred choice"). The text specified in `intro.text` will be displayed before the choice sets. This will generally be a description of the survey and some instructions. The text specified in `end.text` will be displayed after the survey. This will generally be a thanking note and some further instructions.

Value

After completing the survey, two text files can be found in `data.dir`. The file with "num" in the filename is a matrix with the numeric choice data. The coded design matrix ("par"), presented during the survey, together with the observed responses ("resp") can be found here. Rownames indicate the setnumbers. The file with "char" in the filename is a matrix with character choice data. The labeled design matrix ("par"), presented during the survey, together with the observed responses ("resp") can be found here. See `LoadData` to load the data.

References

Crabbe M, Akinc D and Vandebroek M (2014). “Fast algorithms to generate individualized designs for the mixed logit choice model.” http://www.sciencedirect.com/science/article/pii/S0191261513002178.

Examples

```
#### Present choice design without adaptive sets (n.total = sets in des)
# NOTE that the data will be saved in the current working directory.
# example design
data("example_design") # pregenerated design
xdes <- example_design
### settings of the design
code <- c("D", "D", "D")
n.sets <- 8
# settings of the survey
alternatives <- c("Alternative A", "Alternative B")
```

```
attributes <- c("Price", "Time", "Comfort")
labels <- vector(mode="list", length(attributes))
labels[[1]] <- c("$10", "$5", "$1")
labels[[2]] <- c("20 min", "12 min", "3 min")
labels[[3]] <- c("bad", "average", "good")
i.text <- "Welcome, here are some instructions ... good luck!"
b.text <- "Please choose the alternative you prefer"
e.text <- "Thanks for taking the survey"
dataDir <- getwd()
# Display the survey
SurveyApp (des = xdes, n.total = n.sets, alts = alternatives,
          atts = attributes, lvl.names = labels, coding = code,
          buttons.text = b.text, intro.text = i.text, end.text = e.text,
          data.dir = dataDir)
# Data
data_num <- LoadData(data.dir = dataDir, type  = "num")
data_char <- LoadData(data.dir = dataDir, type = "char")

#### Present choice design with adaptive sets (n.total > sets in des)
# NOTE that the data will be saved in the current working directory.
# example design
data("example_design") # pregenerated design
xdes <- example_design
### settings of the design
code <- c("D", "D", "D")
n.sets <- 12
# settings of the survey
alternatives <- c("Alternative A", "Alternative B")
attributes <- c("Price", "Time", "Comfort")
labels <- vector(mode="list", length(attributes))
labels[[1]] <- c("$10", "$5", "$1")
labels[[2]] <- c("20 min", "12 min", "3 min")
labels[[3]] <- c("bad", "average", "good")
i.text <- "Welcome, here are some instructions ... good luck!"
b.text <- "Please choose the alternative you prefer"
e.text <- "Thanks for taking the survey"
# setting for adaptive sets
levels <- c(3, 3, 3)
cand <- Profiles(lvls = levels, coding = code)
p.mean <- c(0.3, 0.7, 0.3, 0.7, 0.3, 0.7)
p.var <- diag(length(p.mean))
dataDir <- getwd()
# Display the survey
SurveyApp (des = NULL, n.total = n.sets, alts = alternatives, atts =
attributes, lvl.names = labels, coding = code, buttons.text = b.text,
intro.text = i.text, end.text = e.text, data.dir = dataDir, crit= "KL",
prior.mean = p.mean, prior.covar = p.var, cand.set = cand, m = 6)
# Data
data_num <- LoadData(data.dir = dataDir, type = "num")
data_char <- LoadData(data.dir = dataDir, type = "char")

#### Choice design with only adaptive sets (des=NULL)
# NOTE that the data will be saved in the current working directory.
# setting for adaptive sets
levels <- c(3, 3, 3)
p.mean <- c(0.3, 0.7, 0.3, 0.7, 0.3, 0.7)
p.var <- diag(length(p.mean))
code <- c("D", "D", "D")
cand <- Profiles(lvls = levels, coding = code)
n.sets <- 12
# settings of the survey
alternatives <- c("Alternative A", "Alternative B")
attributes <- c("Price", "Time", "Comfort")
labels <- vector(mode="list", length(attributes))
labels[[1]] <- c("$10", "$5", "$1")
labels[[2]] <- c("20 min", "12 min", "3 min")
labels[[3]] <- c("bad", "average", "good")
i.text <- "Welcome, here are some instructions ... good luck!"
b.text <- "Please choose the alternative you prefer"
e.text <- "Thanks for taking the survey"
dataDir <- getwd()
# Display the survey
SurveyApp (des = NULL, n.total = n.sets, alts = alternatives,
          atts = attributes, lvl.names = labels, coding = code,
        buttons.text = b.text, intro.text = i.text, end.text = e.text, data.dir = dataDir,
          crit= "KL", prior.mean = p.mean, prior.covar = p.var, cand.set = cand, m = 6)
# Data
data_num <- LoadData(data.dir = dataDir, type = "num")
data_char <- LoadData(data.dir = dataDir, type = "char")
```

Index

∗Topic **data**
aggregate_design, 2
example_design, 6

aggregate_design, 2

Charbin, 2
choicemodelr, *4*
combinations, *12*
contr.sum, *5*, *10*
contr.treatment, *5*, *10*

Datatrans, 3
Decode, 4, *16*
doHB, *4*

example_design, 6

ImpsampMNL, 6, *12*, *14*

LoadData, 8, *16*

Modfed, *6, 7*, 8, *11, 12*, *15*

Profiles, *5*, *8*, 10, *12*, *14, 15*

radioButtons, *2*
rbprobitGibbs, *4*
RespondMNL, *7*, 11
rhierMnlRwMixture, *4*

SeqDB, 12
SeqKL, 13
SurveyApp, *8*, 15