Package 'idefix'

November 9, 2018

Type Package

Title Efficient Designs for Discrete Choice Experiments

Version 0.3.3

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, mlogit, parallel, Rcpp (>= 0.12.18), Rdpack, stats, scales, tmvtnorm, utils

RoxygenNote 6.1.0

Encoding UTF-8

LinkingTo Rcpp, RcppArmadillo

RdMacros Rdpack

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

NeedsCompilation yes

Repository CRAN

Date/Publication 2018-11-09 14:20:08 UTC

R topics documented:

`idefix-package` *idefix: efficient designs for discrete choice experiments.*

Description

Generates efficient designs for discrete choice experiments based on the MNL model, and individually adapted designs for the mixed multinomial logit model. The (adaptive) designs can be presented on screen and choice data can be gathered using a shiny application.

Details

- To generate efficient designs using a modified federov algorithm, please consult the Modfed documentation.
- To generate adaptive designs, please consult the SeqDB documentation.
- To generate a discrete choice survey on screen, please consult the SurveyApp documentation.

Author(s)

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

See Also

Useful links:

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

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

Datatrans *Data transformation.*

Description

Transforms the data into the desired data format required by different estimation packages.

Usage

```
Datatrans(pkg, des, y, n.alts, n.sets, n.resp, bin, alt.names = NULL)
```

Arguments

pkg	Indicates the desired estimation package. Options are bayesm = rhierMnlRwMixture, ChoiceModelR = choicemodelr, RSGHB = doHB, Mixed.Probit = rbprobitGibbs, mlogit = mlogit, and Rchoice = Rchoice).
des	A design matrix in which each row is a profile.
y	A numeric vector containing binary or discrete responses. See bin.
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.
bin	Logical value indicating whether the reponse vector contains binary data (TRUE) or discrete data (FALSE). See y.
alt.names	A character vector containing the names of the alternatives. The default = NULL

Details

The des specified should be the full aggregate design. Thus if all participants responded to the same design, des will be a repetition of that design matrix.

The responses in y should be succesive when there are multiple respondents. There can be n.sets elements for each respondent with discrete values indicating the chosen alternative for each set. Or there can be n.sets * n.alts elements for each respondent with binary values indicating for each alternative whether it was chosen or not. In the latter case the bin argument should be TRUE.

n.sets indicates the number of sets each repondent responded to. It is assumed that every responded responded to the same number of choice sets.

Value

The data ready to be used by the specified package.

Examples

```
idefix.data <- aggregate_design
des <- as.matrix(idefix.data[, 3:8], ncol = 6)
y <- idefix.data[, 9]
bayesm.data <- Datatrans(pkg = "bayesm", des = des, y = y,
n.alts = 2, n.sets = 8, n.resp = 7, bin = TRUE)
Mix.pro.data <- Datatrans(pkg = "Mixed.Probit", des = des, y = y,
 n.alts = 2, n.sets = 8, n.resp = 7, bin = TRUE)
mlogit.data <- Datatrans(pkg = "mlogit", des = des, y = y,
 n.alts = 2, n.sets = 8, n.resp = 7, bin = TRUE)
```

DBerr *DB error*

Description

Function to calculate the DB-error given a design, and parameter values.

Usage

```
DBerr(par.draws, des, n.alts, weights = NULL)
```

Arguments

par.draws	Numeric matrix in which each row is a draw from a multivariate parameter distribution.
des	A design matrix in which each row is an alternative.
n.alts	Numeric value indicating the number of alternatives per choice set.
weights	A numeric vector containing weights of par.draws. The default is NULL.

Value

Numeric value indicating the DB-error of the design given the parameter draws.

Examples

```
des <- example_design
mu = c(-1, -1.5, -1, -1.5, 0.5, 1)
Sigma = diag(length(mu))
par.draws <- MASS::mvrnorm(100, mu = mu, Sigma = Sigma)
n.alts = 2
DBerr(par.draws = par.draws, des = des, n.alts = n.alts)

mu = c(-0.5, -1, -0.5, -1, 0.5, 1)
Sigma = diag(length(mu))
par.draws <- MASS::mvrnorm(100, mu = mu, Sigma = Sigma)
DBerr(par.draws = par.draws, des = des, n.alts = n.alts)
```

Decode *Coded design to readable design.*

Description

Transforms a coded design matrix into a design containing character attribute levels, ready to be used in a survey. The frequency of each attribute level in the design is also included in the output.

Usage

```
Decode(des, n.alts, lvl.names, coding, alt.cte = NULL, c.lvls = NULL,
  no.choice = NULL)
```

Arguments

`des`	A numeric matrix which represents the design matrix. Each row is a profile.
`n.alts`	Numeric value indicating the number of alternatives per choice set.
`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`.
`no.choice`	An integer indicating the no choice alternative. The default is `NULL`.

Details

des A design matrix, this can also be a single choice set. See for example the output of Modfed.

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 of that attribute 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 attribute level is used as reference level.

If des contains columns for alternative specific constants, alt.cte should be specified. In this case, the first column(s) (equal to the number of nonzero elements in alt.cte) will be removed from des before decoding the alternatives.

Value

design	A character matrix which represents the design.
lvl.balance	A list containing the frequency of appearance of each attribute level in the design.

Examples

```
# Example without continuous attributes.
design <- example_design
c <- c("D", "D", "D") # Coding.
# Levels as they should appear in survey.
al <- list(
  c("$50", "$75", "$100"), # Levels attribute 1.
  c("2 min", "15 min", "30 min"), # Levels attribute 2.
  c("bad", "moderate", "good") # Levels attribute 3.
)
# Decode
Decode(des = design, lvl.names = al, coding = c)

# Example with alternative specific constants
design <- example_design2
c <- c("D", "D", "D") # Coding.
# Levels as they should appear in survey.
al <- list(
  c("$50", "$75", "$100"), # Levels attribute 1.
  c("2 min", "15 min", "30 min"), # Levels attribute 2.
  c("bad", "moderate", "good") # Levels attribute 3.
)
# Decode
Decode(des = design, lvl.names = al, coding = c, alt.cte = c(1, 1, 0))
```

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 columns.

example_design2 *Discrete choice design.*

Description

This discrete choice design is generated using the Modfed function. There are 8 choice sets, each containig 3 alternatives (rows). The alternatives consist of 3 attributes (time, price, comfort) with each 3 levels, all of which are dummy coded (columns). The first two colums are alternative specific constants for alternative 1 and 2.

Usage

```
data(example_design2)
```

Format

A matrix with 24 rows and 8 columns.

ImpsampMNL *Importance sampling MNL*

Description

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

Usage

```
ImpsampMNL(n.draws, prior.mean, prior.covar, des, n.alts, y,
  alt.cte = NULL, lower = NULL, upper = NULL)
```

Arguments

n.draws	numeric value indicating the number of draws.
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. If alternative specific constants are present, those should be included as the first column(s) of the design. 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.
alt.cte	A binary vector indicating for each alternative whether an alternative specific constant is desired. The default is NULL.
lower	Numeric vector of lower truncation points, the default is NULL.
upper	Numeric vector of upper truncation points, the default is NULL.

Details

For the proposal distribution a t-distribution with degrees of freedom equal to the number of parameters is used. The mode is estimated using optim, and the covariance matrix is calculated as the negative inverse of the generalized Fisher information matrix. See reference for more information.

From this distribution a lattice grid of draws is generated.

If truncation is present, incorrect draws are rejected and new ones are generated untill n.draws is reached. The covariance matrix is in this case still calculated as if no truncation was present.

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.

References

Yu J, Goos P, 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

```
## Example 1: sample from posterior, no constraints, no alternative specific constants
# choice design
design <- example_design
# Respons.
truePar <- c(0.7, 0.6, 0.5, -0.5, -0.7, 1.7) # some values
set.seed(123)
resp <- RespondMNL(par = truePar, des = design, n.alts = 2)
#prior
pm <- c(1, 1, 1, -1, -1, 1) # mean vector
pc <- diag(1, ncol(design)) # covariance matrix
# draws from posterior.
ImpsampMNL(n.draws = 100, prior.mean =  pm, prior.covar = pc,
           des = design, n.alts = 2, y = resp)

## example 2:  sample from posterior with constraints
# and alternative specific constants
# choice design.
design <- example_design2
# Respons.
truePar <- c(0.2, 0.8, 0.7, 0.6, 0.5, -0.5, -0.7, 1.7) # some values
set.seed(123)
resp <- RespondMNL(par = truePar, des = design, n.alts = 3)
# prior
pm <- c(1, 1, 1, 1, 1, -1, -1, 1) # mean vector
pc <- diag(1, ncol(design)) # covariance matrix
low = c(-Inf, -Inf, 0, 0, 0, -Inf, -Inf, 0)
up = c(Inf, Inf, Inf, Inf, Inf, 0, 0, Inf)
# draws from posterior.
ImpsampMNL(n.draws = 100, prior.mean =  pm, prior.covar = pc, des = design,
           n.alts = 3, y = resp, lower = low, upper = up, alt.cte = c(1, 1, 0))
```

`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 start design with candidate profiles. By doing this it tries to minimize the D(B)-error, based on a multinomial logit model. This routine is repeated for multiple starting designs.

Usage

```
Modfed(cand.set, n.sets, n.alts, par.draws, alt.cte = NULL,
  no.choice = FALSE, start.des = NULL, parallel = TRUE,
  max.iter = Inf, n.start = 12, best = 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.sets	Numeric value indicating the number of choice sets.
n.alts	Numeric value indicating the number of alternatives per choice set.
par.draws	A matrix or a list, dependend on alt.cte.
alt.cte	A binary vector indicating for each alternative whether an alternative specific constant is desired. The default is NULL.
no.choice	A logical value indicating whether a no choice alternative should be added to each choice set. The default is NULL.
start.des	A list containing one or more matrices. The default is NULL.
parallel	Logical value indicating whether computations should be done over multiple cores. The default is TRUE.
max.iter	A numeric value indicating the maximum number allowed iterations. The default is TRUE.
n.start	A numeric value indicating the number of random start designs to use. The default is 12.
best	A logical value indicating whether only the best design should be returned. The default is TRUE.

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. Whenever there are alternative specific constants, `par.draws` should be a list containing two matrices. The first matrix containing the parameter draws for the alternative specific constant parameters. The second matrix containing the draws for the rest of the 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 length of this binary vector should equal `n.alts`, were `0` indicates the absence of an alternative specific constant and `1` the opposite.

`start.des` is a list with one or several matrices. In each matrix 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` + the number of non-zero elements in `alt.cte`. If `start.des = NULL`, `n.start` random start designs will be generated. If start designs are provided, `n.start` is ignored.

If `no.choice` is `TRUE`, in each choice set an alternative with one alternative specific constant is added. The return value of the D(B)-error is however based on the design without the no choice option.

When `parallel` is `TRUE`, `detectCores` will be used to decide upon the number of available cores. That number minus 1 cores will be used to search for efficient designs. The computation time will decrease significantly when `parallel = TRUE`.

Value

If `best = TRUE` the design with the lowest D(B)-error. If `best = FALSE`, the result of all (provided) start designs.

`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`.
`probs`	Numeric matrix containing the probabilities of each alternative in each choice set. If a sample matrix was provided in `par.draws`, this is the average over all draws.

References

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

Examples

```
# DB-efficient designs
# 3 Attributes, all dummy coded. 1 alternative specific constant. = 7 parameters
cand.set <- Profiles(lvls = c(3, 3, 3), coding = c("D", "D", "D"))
mu <- c(0.5, 0.8, 0.2, -0.3, -1.2, 1.6, 2.2) # Prior parameter vector
v <- diag(length(mu)) # Prior variance.
set.seed(123)
pd <- MASS::mvrnorm(n = 10, mu = mu, Sigma = v) # 10 draws.
p.d <- list(matrix(pd[,1], ncol = 1), pd[,2:7])
Modfed(cand.set = cand.set, n.sets = 8, n.alts = 2,
       alt.cte = c(1, 0), parallel = FALSE, par.draws = p.d, best = FALSE)

# DB-efficient design with start design provided.
# 3 Attributes with 3 levels, all dummy coded (= 6 parameters).
cand.set <- Profiles(lvls = c(3, 3, 3), coding = c("D", "D", "D"))
mu <- c(0.8, 0.2, -0.3, -0.2, 0.7, 0.4) # Prior mean (total = 5 parameters).
v <- diag(length(mu)) # Prior variance.
sd <- list(example_design)
set.seed(123)
ps <- MASS::mvrnorm(n = 10, mu = mu, Sigma = v) # 10 draws.
Modfed(cand.set = cand.set, n.sets = 8, n.alts = 2,
       alt.cte = c(0, 0), parallel = FALSE, par.draws = ps, start.des = sd)
```

---

nochoice_design *Discrete choice design with no choice option.*

Description

This discrete choice design is generated using the `Modfed` function. There are 8 choice sets, each containig 3 alternatives (rows), of which one is a no choice option. The no choice option consist of an alternative specific constant and zero's for all other attributelevels. There are three attributes (time, price, comfort) with each 3 levels, all of which are dummy coded (columns).

Usage

```
data(nochoice_design)
```

Format

A matrix with 24 rows and 7 variables

`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. If alternative specific constants are present, those should be included as the first column(s) of the design. 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

```
# design: 3 dummy coded attributes, each 3 levels. There are 8 choice sets.
des <- example_design
set.seed(123)
true_par <- rnorm(6)
RespondMNL(par=true_par, 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 = NULL, cand.set, n.alts, par.draws, prior.covar,
  alt.cte = NULL, no.choice = NULL, weights = NULL,
  parallel = TRUE, reduce = TRUE)
```

Arguments

des	A design matrix in which each row is a profile. If alternative specific constants are present, those should be included as the first column(s) of the design. 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 or a list, dependend on alt.cte.
prior.covar	Covariance matrix of the prior distribution.
alt.cte	A binary vector indicating for each alternative whether an alternative specific constant is desired. The default is NULL.
no.choice	An integer indicating the no choice alternative. The default is NULL.
weights	A vector containing the weights of the draws. Default is NULL, See also ImpsampMNL.
parallel	Logical value indicating whether computations should be done over multiple cores.
reduce	Logical value indicating whether the candidate set should be reduced or not.

Details

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

Previously generated choice sets, which together form an initial design, can be provided in des. When no design is provided, the algorithm will select te most efficient choice set based on the fisher information of the prior covariance matrix prior.covar.

If alt.cte = NULL, par.draws should be a matrix in which each row is a sample from the multivariate parameter distribution. In case that alt.cte is not NULL, a list containing two matrices should be provided to par.draws. The first matrix containing the parameter draws for the alternative specific parameters. The second matrix containing the draws for the rest of the parameters.

The list of potential choice sets are created using combinations. If reduce is TRUE, repeats.allowed = FALSE and vice versa. Furthermore, the list of potential choice sets will be screaned in order to select only those choice sets with a unique information matrix. If no alternative specific 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 argument can be used when the par.draws have weights. This is for example the case when parameter values are updated using ImpsampMNL.

When parallel is TRUE, detectCores will be used to decide upon the number of available cores. That number minus 1 cores will be used to search for the optimal choice set. For small problems (6 parameters), parallel = TRUE can be slower. For larger problems the computation time will decrease significantly.

Value

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

References

Yu J, Goos P, 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, 3), coding = c("E", "E", "E"))
m <- c(0.3, 0.2, -0.3, -0.2, 1.1, 2.4) # mean (total = 6 parameters).
pc <- diag(length(m)) # covariance matrix
set.seed(123)
sample <- MASS::mvrnorm(n = 10, mu = m, Sigma = pc)
# Initial design.
des <- example_design
# Efficient choice set to add.
SeqDB(des = des, cand.set = cs, n.alts = 2, par.draws = sample,
          prior.covar = pc, parallel = FALSE)

# DB efficient choice set, given parameter draws.
# with alternative specific constants
des <- example_design2
cs <- Profiles(lvls = c(3, 3, 3), coding = c("E", "E", "E"))
ac <- c(1, 1, 0) # Alternative specific constants.
m <- c(0.3, 0.2, -0.3, -0.2, 1.1, 2.4, 1.8, 1.2) # mean
pc <- diag(length(m)) # covariance matrix
pos <- MASS::mvrnorm(n = 10, mu = m, Sigma = pc)
sample <- list(pos[ , 1:2], pos[ , 3:8])
# Efficient choice set.
SeqDB(des = des, cand.set = cs, n.alts = 3, par.draws = sample, alt.cte = ac,
          prior.covar = pc, parallel = FALSE)
```

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,
  alt.cte = NULL, no.choice = NULL, buttons.text, intro.text, end.text,
  data.dir = NULL, c.lvls = NULL, prior.mean = NULL,
  prior.covar = NULL, cand.set = NULL, n.draws = NULL,
  lower = NULL, upper = NULL, parallel = TRUE, reduce = TRUE)
```

Arguments

des	A numeric matrix which represents the design matrix. Each row is a profile.
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.
alt.cte	A binary vector indicating for each alternative if an alternative specific constant is present. The default is NULL.
no.choice	An integer indicating which alternative should be a no choice alternative. The default is NULL.
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.
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.
n.draws	numeric value indicating the number of draws.
lower	Numeric vector of lower truncation points, the default is NULL.
upper	Numeric vector of upper truncation points, the default is NULL.
parallel	Logical value indicating whether computations should be done over multiple cores. The default is TRUE.
reduce	Logical value indicating whether the candidate set should be reduced or not.

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. See SeqDB for more information on adaptive choice sets.

Whenever adaptive sets will be generated, prior.mean, prior.covar, cand.set and n.draws, should be specified. These arguments are necesarry for the underlying importance sampling algorithm to update the prior preference distribution. lower and upper can be used to specify lower and upper truncation points. See ImpsampMNL for more details.

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 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.

A no choice alternative is coded as an alternative with 1 alternative specific constant and zero's for all other attribute levels. If a no choice alternative is present in des, or is desired when generating adaptive choice sets, no.choice should be specified. This should be done with an integer, indicating which alternative is the no choice option. This alternative will not be presented on screen, but the option to select "no choice" will be. The alt.cte argument should be specified accordingly, namely with a 1 on the location of the no.choice option. See examples for an example.

When parallel is TRUE, detectCores will be used to decide upon the number of available cores. That number minus 1 cores will be used to search for the optimal adaptive choice set. For small problems (6 parameters), parallel = TRUE can be slower. For larger problems the computation time will decrease significantly.

When reduce = TRUE, the set of all potential choice sets will be reduced to choice sets that have a unique information matrix. If no alternative specific 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.

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

Yu J, Goos P, 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

```
#### Present choice design without adaptive sets (n.total = sets in des)
# 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)

#### Present choice design with partly adaptive sets (n.total > sets in des)
# 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 = 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, crit= "DB", prior.mean = p.mean,
prior.covar = p.var, cand.set = cand, n = 50)

#### Choice design with only adaptive sets (des=NULL)
# setting for adaptive sets
levels <- c(3, 3, 3)
p.mean <- c(0.3, 0.7, 0.3, 0.7, 0.3, 0.7)
low = c(-Inf, -Inf, -Inf, 0, 0, -Inf)
up = rep(Inf, length(p.mean))
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, crit= "KL", prior.mean = p.mean,
prior.covar = p.var, cand.set = cand, lower = low, upper = up, n = 50)


#### Present choice design with a no choice alternative.
# example design
data("nochoice_design") # pregenerated design
xdes <- nochoice_design
### settings of the design
code <- c("D", "D", "D")
n.sets <- 8
# settings of the survey
alternatives <- c("Alternative A", "Alternative B", "None")
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"
```

```
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,
          no.choice = 3, alt.cte = c(0 , 0, 1))
```

Index