Package 'support.CEs'

August 27, 2015

Type Package

Title Basic Functions for Supporting an Implementation of Choice Experiments

Version 0.4-1

Date 2015-08-27

Author Hideo Aizaki

Maintainer Hideo Aizaki <azk-r@spa.nifty.com>

Description Provides seven basic functions that support an implementation of choice experiments.

License GPL (>= 2)

Depends DoE.base, MASS, simex, stats, RCurl, XML

Suggests survival

NeedsCompilation no

Repository CRAN

Date/Publication 2015-08-27 08:31:59

R topics documented:

`support.CEs-package` *Basic functions for supporting an implementation of choice experiments*

Description

The package **support.CEs** provides seven basic functions that support an implementation of choice experiments. These include the following functions: two for creating a choice experiment design, which is based on orthogonal main-effect arrays; one for converting a choice experiment design into questionnaire format; one for converting a choice experiment design into a design matrix; one for making the data set suitable for a conditional logit model analysis using the function `clogit` in the package **survival**, or for a binary choice model analysis using the function `glm` in the package **stats**; one for calculating goodness-of-fit measures for an estimated model; and one for calculating the marginal willingness to pay for the attributes and/or levels of the estimated model. Version 0.3-0 and later versions of this package are also available for binary choice experiments.

Details

The terms in this manual are defined as follows. An "attribute" is a characteristic or feature of an alternative. A "level" or "attribute level" represents the value of an attribute. One attribute can have two or more levels. An "alternative" is a combination of attributes; that is, one alternative can have two or more attributes. For example, when applying choice experiments to marketing research, the alternatives would refer to the "goods" or "services" that respondents are asked to select. A "choice set" refers to a set of alternatives available to individuals. One choice set includes two or more alternatives, including an opt-out alternative, if one exists. In a choice experiment question, respondents are usually asked to select the most preferred alternative from a choice set; therefore, one choice set constitutes a choice experiment question. A "choice experiment design" refers to a collection of individual choice sets.

The following shows an example of a choice experiment design. The choice experiment design includes a total of 9 choice sets (Q1 to Q9). Each choice set (question) consists of three alternatives ("Alternative 1," "Alternative 2," and "None of these" option). "Alternative 1" and "Alternative 2" each consist of three attributes: an attribute A with the three levels of "a1," "a2," and "a3"; an attribute B with the three levels of "b1," "b2," and "b3"; and an attribute C with the three levels of "c1," "c2," and "c3."

Q1. Please select your most preferred alternative from the following:

	Alternative 1	Alternative 2
Attribute A	a2	a3
Attribute B	b2	b3
Attribute C	c2	c3

- I select alternative 1.
- I select alternative 2.
- I select none of these.

Q2. Please select your most preferred alternative from the following:

	Alternative 1	Alternative 2
Attribute A	a2	a1
Attribute B	b3	b2
Attribute C	c1	c3

- I select alternative 1.
- I select alternative 2.
- I select none of these.

« Q3 to Q8 are omitted for the sake of simplicity»

Q9. Please select your most preferred alternative from the following:

	Alternative 1	Alternative 2
Attribute A	a3	a2
Attribute B	b2	b2
Attribute C	c1	c2

- I select alternative 1.
- I select alternative 2.
- I select none of these.

Although there are three alternatives in the example above, the case of two alternatives for each question also exists. Such a choice experiment is known as a binary choice experiment.

Author(s)

Hideo Aizaki

References

Aizaki, H. (2012) Basic Functions for Supporting an Implementation of Choice Experiments in R. *Journal of Statistical Software, Code Snippets*, **50**(2), 1–24. http://www.jstatsoft.org/v50/c02/

See Also

oa.design, clogit, glm

---

gofm *Calculating goodness-of-fit measures*

Description

This function provides rho-squared and rho-squared adjusted by the number of estimated coefficients.

Usage

```
gofm(output)

## S3 method for class 'gofm'
print(x, digits = getOption("digits"), ...)
```

Arguments

output	An object containing the output from the function clogit or glm.
x	An object of class "gofm."
digits	A number of significant digits.
...	Arguments passed to the function format.

Details

This function provides rho-squared (also called McFadden's R-squared or pseudo R-squared), rho-squared adjusted by the number of estimated coefficients, the number of estimated coefficients, and log likelihood values at the start and at convergence.

In version 0.3-0 and later versions, this function is also available for calculating goodness-of-fit measures for binary choice models estimated by using the function glm in the package **stats**.

Value

This function returns an object of S3 class "gofm" that is a list with the following components.

RHO2	The rho-squared value. Defined as RHO2 = 1 - (LLb / LL0).
AdjRHO2	The rho-squared value adjusted by the number of estimated coefficients. Defined as AdjRHO2 = 1 - ((LLb - K) / LL0).

AIC	The Akaike Information Criterion (AIC).
BIC	The Bayesian Information Criterion.
K	The number of estimated coefficients.
LL0	The log likelihood value at the start.
LLb	The log likelihood value at convergence.

Author(s)

Hideo Aizaki

References

Ben-Akiva, M. and Lerman, S. R. (1985) *Discrete Choice Analysis: Theory and Application to Travel Demand*. The MIT Press.

Cameron, A. C. and Trivedi, P. K. (2005) *Microeconometrics: Methods and Applications*. Cambridge University Press.

Aizaki, H. (2012) Basic Functions for Supporting an Implementation of Choice Experiments in R. *Journal of Statistical Software, Code Snippets*, **50**(2), 1–24. http://www.jstatsoft.org/v50/c02/

See Also

clogit, glm, make.dataset

Examples

```
# See "Examples" for the function make.dataset.
```

Lma.design *Creating a choice experiment design using the L^MA method*

Description

This function creates a choice experiment design according to the *L^MA* method.

Usage

```
Lma.design(candidate.array = NULL, attribute.names,
           nalternatives, nblocks, row.renames = TRUE,
           seed = NULL)

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

Arguments

`candidate.array`	A data frame containing an array created by the user. Normally, when this function is used, this argument does not need to be set by the user.
`attribute.names`	A list of the names of attributes and levels.
`nalternatives`	An integer value describing the number of alternatives per choice set, excluding an opt-out alternative such as a "none of these" or a common base alternative.
`nblocks`	An integer value describing the number of blocks into which a choice experiment design is divided.
`row.renames`	A logical variable describing whether or not the row names of a choice experiment design created by this function are changed. When its value is `TRUE` (default), integer values are assigned to the row names starting from 1. When its value is `FALSE`, the row names are the same as those of an array created by the function `oa.design` (included in the **DoE.base** package) via the function `Lma.design`, or those of an array assigned to the argument `candidate.array` by the user.
`seed`	Seed for a random number generator.
`x`	An object of S3 class "ceds."
`...`	Arguments passed to the function `print`.

Details

The *L^MA* method directly creates a choice experiment design from an orthogonal main-effect array (Johnson et al. 2007). In this method, an orthogonal main-effect array with `M` times `A` columns of `L` level factors is used to create each choice set that contains `M` alternatives of `A` attributes with `L` levels. Each row of the array corresponds to the alternatives of a choice set.

This method creates a labeled type choice experiment design that can contain both generic attributes and alternative-specific attributes: the generic attribute refers to that which is included in all the alternatives; the alternative-specific attribute is that which is included in only one alternative. The reader is referred to chapters 3 and 5 of Louviere et al. (2000) for details about the types of attribute—generic or alternative-specific—and the types of choice experiment design—labeled or unlabeled.

When this function is used, the combination of attributes and attribute levels, the number of alternatives per choice set excluding an opt-out or common base option, and the number of blocks are respectively assigned to the arguments.

The combination of attributes and attribute levels are assigned to the argument `attribute.names` in list format. For example, let's assume that the alternative has three attributes, each of which has three levels: an attribute X with the three levels of x1, x2, and x3; an attribute Y with the three levels of y1, y2, and y3; and an attribute Z with the three levels of 10, 20, and 30. In this case, the argument is set as follows:

```
attribute.names = list(X = c("x1", "x2", "x3"),
Y = c("y1", "y2", "y3"), Z = c("10", "20", "30"))
```

The number of alternatives per choice set is defined by the argument `nalternatives`: the number of alternatives does not include an opt-out option such as a "none of these" or a common base option.

When a large choice experiment design is created (that is, there are numerous choice experiment questions), the respondent may carry a heavy psychological burden in terms of answering the questions: in these cases, the choice experiment design is frequently divided into two or more blocks (subsets) of choice sets (questions), and each respondent is asked to answer one block of questions. The argument nblocks assigns the number of blocks. For example, when the argument nblocks is set to be 3 and the choice experiment design contains 27 individual choice sets (that is, there are 27 choice experiment questions), the choice experiment design is divided into 3 blocks, each of which has 9 individual choice sets (9 choice experiment questions). "Blocking" is performed on the basis of a factor with nblocks levels.

Under default settings, this function uses an orthogonal main-effect array that is automatically produced by the function oa.design in the package **DoE.base** based on the argument attribute.names to create a choice experiment design. However, when there is no array corresponding to the argument attribute.names, the function oa.design returns a full factorial based on the argument attribute.names (See help for the function oa.design in the packge **DoE.base**). On the other hand, when this function does not create a choice experiment design matching the user's requirements, the user might achieve it by assigning an arbitrary (user-defined) array to the argument candidate.array: this function uses the array to create a choice experiment design. When the user-defined array is used, the last column of the array must contain a column for dividing the design based on the argument nblocks. The arguments attribute.names and nblocks must also be assigned according to the array.

The function Lma.design can also be used for creating a binary choice experiment design on the basis of an orthogonal main-effect array by setting the argument nalternatives as 1 for a binary choice experiment with an opt-out or common base option, and 2 for a forced-choice format binary choice experiment.

Value

This function returns an object of S3 class "cedes" that is a list with the following components.

alternatives
: A list of objects, alt.j: the *j*th alternative in each choice set created by this function. Each of alt.j includes a variable BLOCK describing the serial number of blocks, a variable QES describing the serial number of choice experiment questions for each value of the variable BLOCK, a variable ALT describing the serial number of alternatives for each value of the QES variable, and attribute variables corresponding to the argument attribute.names.

candidate
: A candidate array used for creating a choice experiment design, which is generated using the function oa.design in the package **DoE.base** or which the user sets for the argument candidate.array. When nblocks >= 2, the last column in this value (candidate) shows a factor that is used for blocking.

design.information
: Information related to the choice experiment design created by this function, which is used as arguments in post-processing functions, such as the functions questionnaire and make.design.matrix. This list includes objects such as the number of blocks into which the choice experiment design is divided (nblocks), the number of questions per block (nquestions), the number of alternatives per choice set excluding an opt-out or common base option (nalternatives), and the number of attributes per alternative (nattributes).

Messages are frequently shown immediately after executing this function when it works properly. These messages are taken from the function `oa.design` and may be valuable to a user who wishes to define the original array and assign it the argument `candidate.array`.

Author(s)

Hideo Aizaki

References

Johnson, F. R., Kanninen, B., Bingham, M. and \"Ozdemir, S. (2007) Experimental Design for Stated Choice Studies. In B. J. Kanninen (ed), *Valuing Environmental Amenities Using Stated Choice Studies: A Common Sense Approach to Theory and Practice*. pp.159–202. Springer.

Louviere, J. J., Hensher, D. A. and Swait, J. D. (2000) *Stated Choice Methods: Analysis and Application*. Cambridge University Press.

Aizaki, H. (2012) Basic Functions for Supporting an Implementation of Choice Experiments in R. *Journal of Statistical Software, Code Snippets*, **50**(2), 1–24. http://www.jstatsoft.org/v50/c02/

See Also

`rotation.design`, `syn.res2`, `oa.design`

Examples

```
# See the second and third cases in "Example"
# for the function make.dataset.
```

`make.dataset` *Making a data set*

Description

This function makes a data set used for a conditional logit model analysis with the function `clogit` in the package **survival** or for a binary choice model analysis with the function `glm` in the package **stats**.

Usage

```
make.dataset(respondent.dataset, design.matrix,
             choice.indicators, detail = FALSE)
```

Arguments

`respondent.dataset`
: A data frame containing respondents' answers to choice experiment questions.

`design.matrix`
: A data frame containing a design matrix created by the function `make.design.matrix` in this package.

`choice.indicators`
: A vector of variables showing the alternative of which was selected in each choice experiment question.

`detail`
: A logical variable describing whether or not some variables contained in the argument `respondent.dataset` and variables created in this function are stored in a data set produced by this function.

Details

Conditional logit model analyses of responses to choice experiment questions in R can be conducted using the function `clogit` in the package **survival**. When the function is used to analyze the responses to the choice experiment questions, a data set in a special format is needed; each alternative should comprise one row of the data set (see the example in Figure 4 in Aizaki and Nishimura 2008). The function `make.dataset` is able to create such a data set by combining a data set containing information about responses to the choice experiment questions and a data set containing a design matrix related to these questions.

The respondent data set has to be created by the user and is assigned to the argument `respondent.dataset`. The data set, in which each row shows one respondent, has to contain a variable `ID`, corresponding to the respondent's identification number; a variable `BLOCK`, corresponding to the serial number of blocks to which each respondent had been assigned; and response variables, corresponding to the answers to each of the choice experiment questions. If necessary, covariates showing the respondent's individual characteristics such as age and gender are also included in the `respondent.dataset`. Although the names of the response variables and covariates are discretionary, the names of the respondent's identification number variable and block variable must be `ID` and `BLOCK`, respectively.

The names of the response variables are assigned to the argument `choice.indicators`. For example, when the names of the response variables in the respondent data set are q1, q2, q3, and q4, the argument is set as `c("q1", "q2", "q3", "q4")`. The response variables show the serial number of the alternative selected by the respondent for each choice experiment question. The method of assigning the serial number of the alternatives must be the same as that of assigning the `ALT` in the output from the `make.design.matrix`. In other words, each alternative must be assigned an integer value that starts from 1. In the `respondent.dataset`, all variables are automatically treated as covariates, except for the variable `ID`, the variable `BLOCK`, and the response variables assigned by the argument `choice.indicators`.

The design matrix data set created by the function `make.design.matrix` is assigned to the argument `design.matrix`.

It should be noted that the order of the questions in the `respondent.dataset` must be the same as that of the variable `QES` in the design matrix data set that was assigned to the argument `design.matrix`, if the order of choice experiment questions presented to respondents was randomized.

The function `make.dataset` can also be used for making a data set suitable for a binary choice model analysis using the function `glm` in the package **stats**.

Value

In addition to some variables contained in the respondent and design matrix data sets, the data set also contains the following two variables that are used in the functions `clogit` and `glm`:

`STR`	A variable assigned to the argument `strata` in the function `clogit` in order to identify each combination of respondent and question.
`RES`	A logical variable taking on `TRUE` when the alternative is selected and `FALSE` when it is not.

Author(s)

Hideo Aizaki

References

Aizaki, H. and Nishimura, K. (2008) Design and Analysis of Choice Experiments Using R: A Brief Introduction. *Agricultural Information Research*, **17**(2), 86–94. http://dx.doi.org/10.3173/air.17.86

Aizaki, H. (2012) Basic Functions for Supporting an Implementation of Choice Experiments in R. *Journal of Statistical Software, Code Snippets*, **50**(2), 1–24. http://www.jstatsoft.org/v50/c02/

See Also

`make.design.matrix`, `syn.res1`, `syn.res2`, `syn.res3`, `clogit`, `glm`

Examples

```
library(survival)
library(stats)

# Case 1
# Choice experiments using the function rotaion.design.
# See "Details" for the data set syn.res1.

des1 <- rotation.design(
 attribute.names = list(
  Region = c("Reg_A", "Reg_B", "Reg_C"),
  Eco = c("Conv.", "More", "Most"),
  Price = c("1", "1.1", "1.2")),
 nalternatives = 2,
 nblocks = 1,
 row.renames = FALSE,
 randomize = TRUE,
 seed = 987)
des1
questionnaire(choice.experiment.design = des1)
desmat1 <- make.design.matrix(
 choice.experiment.design = des1,
 optout = TRUE,
```

```
 categorical.attributes = c("Region", "Eco"),
 continuous.attributes = c("Price"),
 unlabeled = TRUE)
data(syn.res1)
dataset1 <- make.dataset(
 respondent.dataset = syn.res1,
 choice.indicators =
  c("q1", "q2", "q3", "q4", "q5", "q6", "q7", "q8", "q9"),
 design.matrix = desmat1)
clogout1 <- clogit(RES ~ ASC + Reg_B + Reg_C + More + Most +
 More:F + Most:F + Price + strata(STR), data = dataset1)
clogout1
gofm(clogout1)
mwtp(
 output = clogout1,
 monetary.variables = c("Price"),
 nonmonetary.variables =
  c("Reg_B", "Reg_C", "More", "Most", "More:F", "Most:F"),
 seed = 987)

# Case 2
# Choice experiments using the function Lma.design.
# See "Details" for the data set syn.res2.

des2 <- Lma.design(
 attribute.names = list(
  Eco = c("Conv.", "More", "Most"),
  Price = c("1", "1.1", "1.2")),
 nalternatives = 3,
 nblocks = 2,
 row.renames = FALSE,
 seed = 987)
des2
questionnaire(choice.experiment.design = des2, quote = FALSE)
desmat2 <- make.design.matrix(
 choice.experiment.design = des2,
 optout = TRUE,
 categorical.attributes = c("Eco"),
 continuous.attributes = c("Price"),
 unlabeled = FALSE)
data(syn.res2)
dataset2 <- make.dataset(
 respondent.dataset = syn.res2,
 choice.indicators =
  c("q1", "q2", "q3", "q4", "q5", "q6", "q7", "q8", "q9"),
 design.matrix = desmat2)
clogout2 <- clogit(RES ~ ASC1 + More1 + Most1 + Price1 +
 ASC2 + More2 + Most2 + Price2 + ASC3 + More3 + Most3 + Price3 +
 strata(STR), data = dataset2)
clogout2
gofm(clogout2)
mwtp(
 output = clogout2,
```

```
 monetary.variables = c("Price1", "Price2", "Price3"),
 nonmonetary.variables = list(
  c("More1", "Most1"), c("More2", "Most2"), c("More3", "Most3")),
 seed = 987)

# Case 3
# Binary choice experiments using the function Lma.design.
# See "Details" for the data set syn.res3.

des3 <- Lma.design(
 attribute.names = list(
  Region = c("Reg_A", "Reg_B", "Reg_C"),
  Eco = c("Conv.", "More", "Most"),
  Price = c("1", "1.1", "1.2")),
 nalternatives = 1,
 nblocks = 1,
 row.renames = FALSE,
 seed = 987)
des3
questionnaire(choice.experiment.design = des3, quote = FALSE)
desmat3 <- make.design.matrix(
 choice.experiment.design = des3,
 optout = TRUE,
 categorical.attributes = c("Region", "Eco"),
 continuous.attributes = c("Price"),
 unlabeled = TRUE,
 common = NULL,
 binary = TRUE)
data(syn.res3)
dataset3 <- make.dataset(
 respondent.dataset = syn.res3,
 choice.indicators =
  c("q1", "q2", "q3", "q4", "q5", "q6", "q7", "q8", "q9"),
 design.matrix = desmat3)
blout <- glm(RES ~ Reg_B + Reg_C + More + Most + Price,
 family = binomial(link = logit), data = dataset3)
summary(blout)
gofm(blout)
mwtp(output = blout,
 monetary.variables = c("Price"),
 nonmonetary.variables =
  c("Reg_B", "Reg_C", "More", "Most"),
 seed = 987)
```

---

make.design.matrix *Converting a choice experiment design into a design matrix*

Description

This function converts a choice experiment design created by the function `Lma.design` or `rotation.design` into a design matrix suitable for a conditional logit model analysis with the function `clogit` in the package **survival**, or for a binary choice model analysis with the function `glm` in the package **stats**.

Usage

```
make.design.matrix(choice.experiment.design,
                   optout = TRUE,
                   categorical.attributes = NULL,
                   continuous.attributes = NULL,
                   unlabeled = TRUE,
                   common = NULL,
                   binary = FALSE)
```

Arguments

`choice.experiment.design`
: A data frame containing a choice experiment design created by the function `Lma.design` or `rotation.design`.

`optout`
: A logical variable describing whether or not the opt-out alternative is included in the design matrix created by this function. If `TRUE` (default), the opt-out alternative is included; otherwise it is not.

`categorical.attributes`
: A vector containing the names of attributes treated as categorical independent variables in the analysis.

`continuous.attributes`
: A vector containing the names of attributes treated as continuous independent variables in the analysis.

`unlabeled`
: A logical variable describing the types of a choice experiment design assigned by the argument `choice.experiment.design`. If the type is unlabeled, the argument is set as `TRUE` (default). If the type is labeled, it is set as `FALSE`.

`common`
: A vector containing a fixed combination of attribute-levels corresponding to a common base option in each question. If there is no common base option, the argument is set as `NULL` (default).

`binary`
: When the function is applied to the conditional logit model, the argument is set as `FALSE` (default). When the function is applied to the binary choice model, it is set as `TRUE`.

Details

This function converts a choice experiment design created by the function `Lma.design` or `rotation.design` into a design matrix that is suitable for conditional logit model analysis with the function `clogit` in the package **survival** or binary choice model analysis with the function `glm` in the package **stats**.

A choice experiment design created by the function `Lma.design` or `rotation.design` is assigned to the argument `choice.experiment.design`.

Attributes included in the choice experiment design assigned to the argument `choice.experiment.design` are classified into categorical and continuous attributes that are assigned to the arguments `categorical.attributes` and `continuous.attributes`, respectively. For example, an alternative may have three attributes such as X, Y, and Z. In the conditional logit model analysis, when attributes X and Y are treated as categorical variables and attribute Z is treated as continuous variable, the arguments are set as follows:

```
categorical.attributes = c("X", "Y")
continuous.attributes = c("Z")
```

The categorical variables are created in dummy-variable format. In other words, the minimum value in a categorical attribute is normalized; as a result, each dummy variable is assigned a value of 1 when the categorical attribute takes on a value other than the minimum value. The dummy variables are referred to by their levels. For example, in a categorical attribute X with the three levels of "x1," "x2," and "x3," the dummy variables are created as follows: (1) When the choice experiment design is unlabeled, two dummy variables are created: a dummy variable x2 that assumes a value of 1 when attribute X takes "x2," and 0 otherwise; and a dummy variable x3 that assumes a value of 1 when attribute X takes "x3," and 0 otherwise. (2) When the choice experiment design is labeled and the design contains two alternatives ("alternative 1" and "alternative 2"), excluding an opt-out alternative, four dummy variables are created: a dummy variable x21 that assumes a value of 1 when attribute X in alternative 1 takes "x2," and 0 otherwise; a dummy variable x22 that assumes a value of 1 when attribute X in alternative 2 takes "x2," and 0 otherwise; a dummy variable x31 that assumes a value of 1 when attribute X in alternative 1 takes "x3," and 0 otherwise; and a dummy variable x32 that assumes a value of 1 when attribute X in alternative 2 takes "x3," and 0 otherwise.

Two points should be noted with regard to continuous and categorical variables in the function `make.design.matrix`. First, the level of the argument `continuous.attributes` must take on numerical values: that is, the level must not contain a unit of the attribute, such as "USD," "kg," or "km." For example, when the argument `continuous.attributes` is set as `c("Z")` and it shows the price attribute of a product alternative, the variable Z must not contain the levels `USD10`, `USD20`, and `USD30` but must have the levels of `10`, `20`, and `30`, respectively. Second, categorical variables created by the function are not in factor format. R usually treats categorical variables as factors. However, values of attribute variables in each row corresponding to an opt-out option must be set as zero (`0`) because the systematic component of the utility for the opt-out option is normalized to zero. Therefore, the function `make.design.matrix` converts categorical attributes into dummy variables (the same treatment is applied to the function `make.dataset`).

The argument `unlabeled` is set as `TRUE` when the choice experiment design assigned to the argument `choice.experiment.design` is unlabeled; it is `FALSE` otherwise (when the choice experiment design is labeled).

The argument `optout` is set as `TRUE` when an opt-out alternative such as the option "none of these" is included in the choice experiment questions. It is set as `FALSE` when the opt-out alternative is not included.

When a common base option, which is also known as the constant comparator, is included in each choice set, a combination of attribute-levels corresponding to the common base is assigned to the argument `common`. For example, when the common base is an alternative in which attribute X takes "x1," attribute Y takes "y2" and attriute Z "10," the argument is set as follows:

```
common = c(X = "x1", Y = "y2", Z = "10")
```

The argument `common` is set as `NULL` when the common base option is not included. It is noted that levels of categorical attributes in the common base option are limited to those that are used in the

design assigned to the argument choice.experiment.design.

When this function is used for constructing a design matrix for the function clogit, the argument binary is set as FALSE (default); the argument is set as TRUE when the function is applied to binary choice models with the function glm.

Value

This function provides a design matrix that can be directly assigned to an argument design.matrix in the function make.dataset. The design matrix contains categorical and/or continuous variables created by this function as well as the following four kinds of variables.

BLOCK	An integer variable describing the serial number of blocks.
QES	An integer variable describing the serial number of questions according to the value of the variable BLOCK.
ALT	An integer variable describing the serial number of alternatives according to the value of the variable QES.
ASC	Alternative specific constant(s). When the choice experiment design is labeled, the serial number of alternatives (ALT) is automatically appended to the tail of ASC (such as ASC1, ASC2, and ASC3)

Author(s)

Hideo Aizaki

References

Aizaki, H. and Nishimura, K. (2008) Design and Analysis of Choice Experiments Using R: A Brief Introduction. *Agricultural Information Research*, **17**(2), 86–94. http://dx.doi.org/10.3173/air.17.86

Aizaki, H. (2012) Basic Functions for Supporting an Implementation of Choice Experiments in R. *Journal of Statistical Software, Code Snippets*, **50**(2), 1–24. http://www.jstatsoft.org/v50/c02/

See Also

Lma.design, rotation.design, make.dataset, syn.res1, syn.res2, syn.res3, clogit, glm

Examples

```
# See "Examples" for the function make.dataset.
```

mwtp *Calculating the marginal willingness to pay*

Description

This function calculates the marginal willingness to pay for the attributes and/or levels of the estimated model.

Usage

```
mwtp(output, monetary.variables, nonmonetary.variables = NULL,
     nreplications = 10000,
     percentile.points = NULL,
     confidence.level = 0.95,
     method = "kr",
     seed = NULL)

## S3 method for class 'mwtp'
print(x, digits = max(3, getOption("digits") - 3),
      scientific = FALSE, ...)
```

Arguments

output	An object containing the output from the function clogit in the package **survival** or from the function glm in the package **stats**.
monetary.variables	A vector containing the names of the monetary variables in the output used to calculate the MWTPs.
nonmonetary.variables	A vector containing the names of the non-monetary variables in the output used to calculate the MWTPs. Its default is NULL.
nreplications	An integer value denoting the number of replications in the simulation method. The default value is set as 10000.
percentile.points	It is only kept for giving an error message regarding unused argument. It will be removed. The argument confidence.level is used instead.
confidence.level	A value showing the confidence level (coefficient) of an empirical distribution of each MWTP. Its default vector is set as 0.95, which indicates the lower and upper bounds of the 95 percent confidence interval.
method	A character variable describing the method used for calculating confidence intervals of MWTPs. It is set as "kr" if the Krinsky and Robb's method is used; it is set as "delta" if the delta method is used.
seed	Seed for a random number generator.
x	An object of S3 class "mwtp."

digits	A number of significant digits. See the function format.
scientific	Whether MWTPs and their confidence intervals are encoded in scientific format. See the function format.
...	Arguments passed to the function format.

Details

The definition of the marginal willingness to pay (MWTP) for a non-monetary variable provided by this function is $-b_{nm}/b_m$; where, b_{nm} is the estimated coefficient of the non-monetary variable, and b_m is the estimated coefficient of a monetary variable. Further, confidence intervals for the MWTPs are calculated according to the simulation method proposed by Krinsky and Robb (1987) or the delta method (see, e.g., Hole 2007).

In the Krinsky and Robb's method, *N* replications of a vector of the coefficients in the model are randomly sampled from a multivariate normal distribution with a vector of means and a variance-covariance matrix of the estimated coefficients. An empirical distribution for each of the MWTPs can be generated from *N* sets of the replicated coefficients, and a confidence interval for each of the MWTPs is identified on the basis of each empirical distribution.

In the delta method, a variance of MWTP of the non-monetary variable is calculated using estimated coefficients and variance-covariance matrix regarding the non-monetary and monetary variables, and then a confidence interval for the MWTP is calculated.

When the argument nonmonetary.variables is not set by the user, variables in the argument output—except for those assigned by the argument monetary.variables—are treated as non-monetary variables, and the MWTPs for these variables are calculated. In the model that assumes alternative-specific attribute variables (that is, a labeled type choice experiment design), variables in the argument output are classified into monetary and non-monetary variables according to the alternatives. Therefore, the argument monetary.variables is set as a vector, whereas the argument nonmonetary.variables is set as a list of vectors.

In version 0.3-0 and later versions, this function is also available for binary choice models estimated using the function glm.

Value

This function returns an object of S3 class "mwtp" that is a list with the following components.

mwtp.table	A matrix containing the MWTPs for the non-monetary attribute variables and confidence intervals for each of the MWTPs.
method	A character variable containing the method used for calculating confidence intervals of MWTPs.
mwtps	A matrix containing empirical distributions of the MWTPs. It is included when the Krinsky and Robb's method is used.
repb	A matrix containing N sets of replicated coefficients. It is included when the Krinsky and Robb's method is used.

The object mwtps can be used for a function mded in the package **mded** that calculates differences between two independent/dependent empirical distributions of the MWTPs.

Author(s)

Hideo Aizaki

References

Louviere, J. J., Hensher, D. A. and Swait, J. D. (2000) *Stated Choice Methods: Analysis and Application*. Cambridge University Press.

Krinsky, I. and Robb. A. L. (1986) On Approximating the Statistical Properties of Elasticities. *The Review of Economics and Statistics*, **68**, 715–719.

Hole, A. R. (2007) A Comparison of Approaches to Estimating COnfidence Intervals for Willingness to Pay Measures. *Health Economics*, **16**, 827–840.

Aizaki, H. (2012) Basic Functions for Supporting an Implementation of Choice Experiments in R. *Journal of Statistical Software, Code Snippets*, **50**(2), 1–24. http://www.jstatsoft.org/v50/c02/

See Also

make.dataset, clogit, glm, mded

Examples

```
# See "Examples" for the function make.dataset.
```

---

pork *Synthetic respondent data set: consumers' valuation of pork*

Description

Data set artificially created for an example based on a labeled DCE design. This example illustrates consumers' valuation of pork.

Usage

```
data(pork)
```

Format

Data frames with 200 respondents on the following 6 variables.

`ID` Identification number of respondents.

`BLOCK` Serial number of blocks to which each respondent had been assigned.

`q1` Response to choice experiment question 1.

`q2` Response to choice experiment question 2.

`q3` Response to choice experiment question 3.

`q4` Response to choice experiment question 4.

Author(s)

Hideo Aizaki

See Also

make.dataset, make.design.matrix, Lma.design, clogit

Examples

```
library(survival)

d.pork <- Lma.design(
  attribute.names = list(
    Price = c("100", "130", "160", "190")),
  nalternatives = 3,
  nblocks = 4,
  row.renames = FALSE,
  seed = 987)

data(pork)

dm.pork <- make.design.matrix(
  choice.experiment.design = d.pork,
  optout = TRUE,
  continuous.attributes = c("Price"),
  unlabeled = FALSE)

ds.pork <- make.dataset(
  respondent.dataset = pork,
  choice.indicators =
    c("q1", "q2", "q3", "q4"),
  design.matrix = dm.pork)

fm.pork <- RES ~ ASC1 + Price1 +
                 ASC2 + Price2 +
                 ASC3 + Price3 +
                 strata(STR)
out.pork <- clogit(fm.pork, data = ds.pork)
out.pork
```

---

questionnaire *Converting a choice experiment design into a choice experiment questionnaire*

Description

This function converts a choice experiment design created by the function Lma.design or rotation.design into choice experiment questions used in a questionnaire survey.

Usage

```
questionnaire(choice.experiment.design,
              common = NULL,
              quote = TRUE)
```

Arguments

choice.experiment.design	A data frame containing a choice experiment design created by the function Lma.design or rotation.design.
common	A vector containing a fixed combination of attribute-levels corresponding to a common base option in each question. If there is no common base option, the argument is set as NULL (default).
quote	A logical variable indicating whether or not the attribute-levels in each question are printed wth quotation marks.

Details

This function converts a choice experiment design created by the function Lma.design or rotation.design into choice experiment questions used in a questionnaire survey.

Value

Choice experiment questions converted from the choice experiment design are returned.

Author(s)

Hideo Aizaki

References

Aizaki, H. (2012) Basic Functions for Supporting an Implementation of Choice Experiments in R. *Journal of Statistical Software, Code Snippets*, **50**(2), 1–24. http://www.jstatsoft.org/v50/c02/

See Also

Lma.design, rotation.design

Examples

```
# See "Examples" for the function make.dataset.
```

rice *Synthetic respondent data set: consumers' valuation of rice*

Description

Data set artificially created for an example based on an unlabeled DCE design. This example illustrates consumers' valuation of rice.

Usage

```
data(rice)
```

Format

Data frames with 100 respondents on the following 12 variables.

`ID` Identification number of respondents.

`BLOCK` Serial number of blocks to which each respondent had been assigned.

`q1` Response to choice experiment question 1.

`q2` Response to choice experiment question 2.

`q3` Response to choice experiment question 3.

`q4` Response to choice experiment question 4.

`q5` Response to choice experiment question 5.

`q6` Response to choice experiment question 6.

`q7` Response to choice experiment question 7.

`q8` Response to choice experiment question 8.

`q9` Response to choice experiment question 9.

`F` Female dummy variable (`1` = Female, otherwise `0`).

Author(s)

Hideo Aizaki

See Also

`make.dataset`, `make.design.matrix`, `rotation.design`, `clogit`

Examples

```
library(survival)

d.rice <- rotation.design(
  attribute.names = list(
    Region = c("RegA", "RegB", "RegC"),
    Cultivation = c("Conv", "NoChem", "Organic"),
    Price = c("1700", "2000", "2300")),
  nalternatives = 2,
  nblocks = 1,
  row.renames = FALSE,
  randomize = TRUE,
  seed = 987)

data(rice)

dm.rice <- make.design.matrix(
  choice.experiment.design = d.rice,
  optout = TRUE,
  categorical.attributes = c("Region", "Cultivation"),
  continuous.attributes = c("Price"),
  unlabeled = TRUE)

ds.rice <- make.dataset(
  respondent.dataset = rice,
  choice.indicators =
    c("q1", "q2", "q3", "q4", "q5",
      "q6", "q7", "q8", "q9"),
  design.matrix = dm.rice)

fm.rice <- RES ~ ASC + RegB + RegC + NoChem + Organic +
           NoChem:F + Organic:F + Price + strata(STR)
out.rice <- clogit(fm.rice, data = ds.rice)
out.rice
```

`rotation.design` *Creating a choice experiment design using the rotation or mix-and-match method*

Description

This function creates a choice experiment design according to the rotation or mix-and-match method.

Usage

```
rotation.design(candidate.array = NULL, attribute.names,
                nalternatives, nblocks, row.renames = TRUE,
                randomize = FALSE, seed = NULL)
```

Arguments

`candidate.array`
: A data frame containing an array created by the user. Normally, when this function is used, this argument does not need to be set by the user.

`attribute.names`
: A list of the names of attributes and levels.

`nalternatives`
: An integer value describing the number of alternatives per choice set, excluding an opt-out alternative such as a "none of these" or common base alternative.

`nblocks`
: An integer value describing the number of blocks into which a choice experiment design is divided.

`row.renames`
: A logical variable describing whether or not the row names of a choice experiment design created by this function are changed. When its value is `TRUE` (default), integer values are assigned to the row names starting from 1. When its value is `FALSE`, the row names are the same as those of an array created by the function `oa.design` (included in the package **DoE.base**) via the function `rotation.design`, or those of an array assigned to the argument `candidate.array` by the user.

`randomize`
: If this argument is `TRUE` (default), the function executes the mix-and-match method. If `FALSE`, the function executes the rotation method.

`seed`
: Seed for a random number generator.

Details

This function creates an unlabeled (generic) choice experiment design according to the rotation or mix-and-match method (Johnson et al. 2007) (see "Details" for the function `Lma.design` about the types of choice experiment design). Each method depends on an orthogonal main-effect array generated by the function `oa.design` in the package **DoE.base** or an arbitrary array set by the user.

The rotation method uses an orthogonal main-effect array as the first alternative in each choice set; this method creates one or more additional alternative(s) by adding a constant to each attribute level of the first alternative; the *k*th(>= 2) alternative in the *j*th (= 1, 2, ..., J) choice set is created by adding one to each of the *m* attributes in the *k* - 1 th alternative in the *j*th choice set. If the level of the attribute in the *k* - 1 th alternative is maximum, then the level of the attribute in the *k* th alternative is assigned the minimum value.

The mix-and-match method modifies the rotation method by introducing the randomizing process. After placing a set of *N* alternatives created from the orthogonal main-effect array into an urn, one or more additional set(s) of *N* alternatives are created using the rotation method and placed into different urn(s). A choice set is generated by selecting one alternative from each urn at random. This selection process is repeated, without replacement, until all the alternatives are assigned to *N* choice sets. These *N* choice sets correspond to a choice experiment design.

When the mix-and-match method is implemented by this function, the argument `randomize` is set as `TRUE` (default). When the rotation method is implemented by this function, the argument is set as `FALSE`.

When this function is used, the combination of attributes and attribute levels, the number of alternatives per choice set excluding an opt-out or common base option, and the number of blocks are respectively assigned to the arguments.

The combination of attributes and attribute levels are assigned to the argument attribute.names in list format. For example, the alternative has three attributes, each of which has three levels: an attribute X with the three levels of x1, x2, and x3; an attribute Y with the three levels of y1, y2, and y3; and an attribute Z with the three levels of 10, 20, and 30. In this case, the argument is set as follows:

```
attribute.names = list(X = c("x1", "x2", "x3"),
Y = c("y1", "y2", "y3"), Z = c("10", "20", "30"))
```

The number of alternatives per choice set is defined by the argument nalternatives: the number of alternatives does not include an opt-out option such as a "none of these" or a common base option.

Similar to the function Lma.design, this function can divide a choice experiment design into two or more blocks based on the argument nblocks. A choice experiment design is randomly divided into nblocks blocks; therefore, nblocks must be divisors of the number of choice sets included in the choice experiment design.

Under default settings, this function uses an orthogonal main-effect array that is automatically produced by the function oa.design based on the argument attribute.names to create a choice experiment design. However, when there is no array corresponding to the argument attribute.names, the function oa.design returns a full factorial based on the argument attribute.names (See help for the function oa.design in the packge **DoE.base**). On the other hand, when this function does not create a choice experiment design matching the user's requirements, the user might achieve it by assigning an arbitrary (user-defined) array to the argument candidate.array: this function uses the array to create a choice experiment design. When the user-defined array is used, the last column of the array must contain a column for dividing the design based on the argument nblocks. The arguments attribute.names and nblocks must also be assigned according to the array.

Value

This function returns an object of S3 class "cedes" (see Lma.design) that is a list with the following components.

alternatives
: A list of objects, alt.j: the *j*th alternative in each choice set created by this function. Each of alt.j includes a variable BLOCK describing the serial number of blocks, a variable QES describing the serial number of choice experiment questions for each value of the variable BLOCK, a variable ALT describing the serial number of alternatives for each value of the QES variable, and attribute variables corresponding to the argument attribute.names.

candidate
: A candidate array used for creating a choice experiment design, which is generated using the oa.design function in the **DoE.base** package or which the user sets for the argument candidate.array. When nblocks >= 2, the last column in this value (candidate) shows a factor that is used for blocking.

design.information
: Information related to the choice experiment design created by this function, which is used as arguments in post-processing functions, such as the functions questionnaire and make.design.matrix. This list includes objects such as the number of blocks into which the choice experiment design is divided (nblocks), the number of questions per block (nquestions), the number of alternatives per choice set excluding an opt-out or common base option (nalternatives), and the number of attributes per alternative (nattributes).

An error message is displayed when the argument nblocks does not match a choice experiment design. In such a case, the argument nblocks is set to be a divisior of the number of rows of the choice experiment design. Ohter messages are frequently shown immediately after executing this function when it works properly. These messages are taken from the function oa.design and may be valuable to a user who wishes to define the original array and assign it the argument candidate.array.

Author(s)

Hideo Aizaki

References

Johnson, F. R., Kanninen, B., Bingham, M. and \"Ozdemir, S. (2007) Experimental Design for Stated Choice Studies. In B. J. Kanninen (ed), *Valuing Environmental Amenities Using Stated Choice Studies: A Common Sense Approach to Theory and Practice*. pp.159–202. Springer.

Aizaki, H. (2012) Basic Functions for Supporting an Implementation of Choice Experiments in R. *Journal of Statistical Software, Code Snippets*, **50**(2), 1–24. http://www.jstatsoft.org/v50/c02/

See Also

Lma.design, syn.res1, oa.design

Examples

```
# See "Examples" for the function make.dataset.
```

---

rural — *Synthetic respondent data set: residents' valuation of rural environment conservation plan*

Description

Data set artificially created for an example based on a BDCE design. This example illustrates residents' valuation of rural environment conservation plan.

Usage

```
data(rural)
```

Format

Data frames with 400 respondents on the following 7 variables.

ID Identification number of respondents.

BLOCK Serial number of blocks to which each respondent had been assigned.

q1 Response to choice experiment question 1.

`q2` Response to choice experiment question 2.

`q3` Response to choice experiment question 3.

`q4` Response to choice experiment question 4.

`Region` Region variable denoting whether the respondent was sampled from region 1 (`Region = 1`) or region 2 (`Region = 2`).

Author(s)

Hideo Aizaki

See Also

`make.dataset`, `make.design.matrix`, `Lma.design`, `glm`

Examples

```
library(stats)

d.rural <- Lma.design(
  attribute.names = list(
    Area = c("20", "40", "60", "80"),
    Facility = c("None", "Agr", "Env", "Rec"),
    Tax = c("1000", "3000", "5000", "7000")),
  nalternatives = 1,
  nblocks = 4,
  row.renames = FALSE,
  seed = 987)

common.alt <- c(Area = "0", Facility = "None", Tax = "0")

dm.rural <- make.design.matrix(
  choice.experiment.design = d.rural,
  optout = FALSE,
  categorical.attributes = c("Facility"),
  continuous.attributes = c("Area", "Tax"),
  unlabeled = TRUE,
  common = common.alt,
  binary = TRUE)

data(rural)
rural1 <- subset(rural, Region == 1)
rural2 <- subset(rural, Region == 2)

ds.rural1 <- make.dataset(
  respondent.dataset = rural1,
  choice.indicators =
    c("q1", "q2", "q3", "q4"),
  design.matrix = dm.rural,
  detail = FALSE)

ds.rural2 <- make.dataset(
```

```
  respondent.dataset = rural2,
  choice.indicators =
    c("q1", "q2", "q3", "q4"),
  design.matrix = dm.rural,
  detail = FALSE)

fm.rural <- RES ~ Agr + Env + Rec + Area + Tax

out.rural1 <- glm(fm.rural,
                  family = binomial(link = "logit"),
                  data = ds.rural1)
summary(out.rural1)

out.rural2 <- glm(fm.rural,
                  family = binomial(link = "logit"),
                  data = ds.rural2)
summary(out.rural2)
```

`sb.design` *Creating or checking a choice experiment design using the Street and Burgess Method*

Description

This function creates or checks a choice experiment design according to the method developed by Street, D. J. and Burgess, L. on the website **Discrete Choice Experiments**.

Usage

```
sb.design(operation = "construct", nattributes, nalternatives, nlevels,
          attribute.names, design = NULL, generators = NULL,
          effect = "main", interactions = NULL, determinant = NULL,
          nblocks = 1, seed = NULL, ...)

## S3 method for class 'sb'
print(x, ...)

## S3 method for class 'sb'
summary(object, ...)

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

Arguments

`operation`	A character variable describing the operation to be conducted: `"construct"` is used to construct a choice experiment design; `"check"` is used to check the choice experiment design that is assigned to the argument `design`.

`nattributes` An integer value describing the number of attributes per alternative included in the choice sets. The range must be 2 <= `nattributes` <= 20.

`nalternatives` An integer value describing the number of alternatives (i.e., options) per choice set included in the orthogonal main effect design (OMED) or choice sets, excluding an opt-out alternative or a common base alternative. The range must be 2 <= `nalternatives` <= 20.

`nlevels` An integer vector describing number of levels for each attribute included in the choice sets. The values must be in the range [2, 20]. The order of the values must correspond to the order of attributes shown in an OMED or choice sets assigned to the argument `design`. For example, when `nlevels = c(2, 2, 3, 4)` is assigned to this argument, this means that the first and second attributes for the OMED/choice sets both have two levels, while the third and fourth attributes have three and four levels, respectively.

`attribute.names`

A list of the names of attributes and levels. Levels must be assigned as a character vector (see below). This argument is optional. The number of attributes included in this argument must be equal to the argument `nattributes`. The number of levels for each attribute must be equal to the values in the argument `nlevels`, and they must be in the same order. Consider the following example: attribute A has two levels, a0 and a1; attribute B has two levels, b0 and b1; attribute C has three levels, c0, c1, and c2; and attribute D has four levels, d0, d1, d2, and d3. These four attributes are assigned to the argument as follows:
`attribute.names = list(A = c("a0", "a1"), B = c("b0", "b1"), C = c("c0", "c1", "c2"), D = c("d0", "d1", "d2", "d3"))`.
In this case, the first, second, third, and fourth element of `nlevels` must show the number of levels for attributes A, B, C, and D, respectively:
`nlevels = c(2, 2, 3, 4)`.

`design` A matrix describing an OMED corresponding to the first alternative for the argument `operation = "construct"` or choice sets for the argument `operation = "check"`. For an OMED with the argument `operation = "construct"`, each row corresponds to a treatment combination (i.e., an alternative), and each column corresponds to an attribute. For the choice sets with the argument `operation = "check"`, each row corresponds to a choice set, and each `nattributes` column corresponds to the attributes for each alternative. The first `nattributes` column correspond to the `nattributes` attributes for the first alternative, the second `nattributes` column corresponds to the `nattributes` attributes for the second alternative, and so on. For either the OMED or choice sets, the values in each column of design must be integers between 0 and (`nlevels` - 1), and they must correspond to the column. When generating an OMED with this function, the argument `design` is set to `NULL`. Thus, an OMED is generated according to the vector assigned to the argument `nlevels` using the function `oa.design()` in the **DoE.base** package.

`generators` An integer vector or matrix describing the sets of generators for constructing the choice sets. Each row (or a vector) corresponds to one set of generators. One set of generators consists of (`nalternative` - 1) * `nattributes` elements. For example, assume the construction of choice sets that have three alternatives each, where each alternative has four attributes. A single set of generators thus

consists of eight integer values (= (3 - 1) * 4). The first four elements correspond to the generator for constructing the second alternative in the choice sets. The remaining four elements correspond to the generator for constructing the third alternative in the choice sets. When only a single set of generators is selected, it is assigned to the argument `generators` as an integer vector. When two or more sets of generators are selected, they are assigned to the argument `generators` as an integer matrix. A set of generators with only elements of 0 are not permitted. For details regarding the formatting style of the generators, refer to section **Examples**, below.

`effect` A character variable describing the effect to be estimated: `"main"` is used for main effects; `"mplusall"` is used to estimate the main effects as well as all of the two-factor interactions; and `"mplussome"` is used to estimate the main effects along with only some two-factor interactions.

`interactions` A vector or list of vectors describing the two-factor (i.e., two-attribute) interactions to be estimated. A single two-factor interaction is assigned as a vector with two elements. For example, `c(1, 2)` denotes the interaction between the first attribute and the second attribute. Two or more of these two-factor interactions are assigned as a list of two or more vectors with two elements each. For example, `list(c(1, 2), c(1, 3))` denotes two different two-factor interactions, one between the first attribute and the second attribute, and the other between the first attribute and the third attribute. The paired values must differ. The values must be in the range [1, `nattributes`]. This argument is valid only when the argument `effect` is set as `"mplussome"`.

`determinant` A character variable describing the determinant of the information matrix. This argument is optional. The determinant must be in the range [0, 1]. The formatting style can be, e.g., `"17/12524124635136"` or `"1.35738e-12"`. Note that the determinant is not assigned as a numerical value (`17/12524124635136` or `1.35738e-12`), but rather as a character (`"17/12524124635136"` or `"1.35738e-12"`). In other words, it is important to enclose the determinant within double quotation marks.

`nblocks` An integer value describing the number of blocks into which a choice experiment design is divided. This option is valid only when `operation ="construct"`. Note that one way to divide a design used in this function is simply to divide the design that is constructed by the website **Discrete Choice Experiments** into `nblocks` subsets randomly. In that way, features of the design such as efficiency are not considered when dividing it.

`seed` A seed for a random number generator.

`...` Optional arguments.

`x, object` Object of the S3 class `"sb"`.

Details

The website **Discrete Choice Experiments** (Burgess 2007) allows us to construct and check choice experiment designs according to the theory explained in Street and Burgess (2007). This function sends a request to the website to construct or check a choice experiment design from R. The calculations for constructing and checking are executed on the website. Thus, this function is merely an

interface between your R and the website. When using this function, your R must be connected to the Internet.

The Street and Burgess Method constructs optimal choice experiment designs that are designed to estimate the main effects and two-factor interaction effects. The first alternative (attribute-levels are expressed as integer values) is derived from an orthogonal main effect design, and the j-th (j = 2, 3, ...) alternative is created by adding special combinations of integer values (i.e., generators) to the first alternative. Refer to Street and Burgess (2007) for more details.

Among the arguments in this function, the following are important: `operation`, `nattributes`, `nlevels`, `nalternatives`, `design`, `generators`, `effect`, `interactions`, and `determinant`. Explanations for these arguments above are based on the **Help** page on website **Discrete Choice Experiments** (Burgess 2007): please refer to the **Help** page for more information regarding these arguments.

Note that a message `"Error in sb.check.args(..."` will appear on the R console after executing this function with incorrect settings for the arguments. In such a case, set the argument correctly according to the message.

Value

This function returns an object of the S3 class `"sb"` inheriting from the S3 class `"cedes"`. This object is a list with `alternatives`, `candidate`, `design.information`, and `sb`. The components `alternatives`, `candidate`, and `design.information` are the same as those in the S3 class `"cedes"` (see `Lma.design` or `rotation.design` for details). When `operation = "check"`, each of `alternatives`, `candidate`, and `design.information` is `NULL`: the output from this function cannot be used in other functions (i.e., `questionnaire` and `make.design.matrix`) in the package **support.CEs** when `operation = "check"`.

The component `sb`, which is the output from the website **Discrete Choice Experiments**, is a list with the following six components:

`input`	A list of objects that were entered: `operation`, `effect`, `design` (treatment combinations or choice sets), `generators`, `determinant`, and `interactions`.
`message`	A character string containing messages regarding the calculation on the website.
`output`	A list of the resulting matrices: the choice sets, the contrast matrix (B), the Lambda matrix, the information matrix (C), the inverse of C, and the correlation matrix.
`calculation`	A logical variable taking on `TRUE` when the calculation process on the website is success, `FALSE` otherwise.
`html`	A character string containing html content received from the website. The length of the character string is long.
`version`	A character string containing software version used for the calculation process on the website.

Note that the order of rows in the design contained in the component `alternatives` is different from that in `sb` when `nblocks` >= 2. This is because the random division of the design into `nblocks` subset is executed after the design is constructed on the website.

This function may return a list of objects including html content when the Street and Burgess Method is not be implemented successfully. The html content is received from the website **Discrete Choice Experiments**. By investigating this content, it is possible to determine what caused the unsuccessful results.

Acknowledgement

I would like to thank Dr. Burgess for her kind support in developing this function.

Note

When publishing results obtained using this function, please cite Burgess (2007), Street and Burgess (2007), and the package **support.CEs**. When creating and publishing new functions that access the website **Discrete Choice Experiments** on the basis of this function, please inform Dr. Burgess about the publication. Dr. Burgess' contact information is provided on the website **Discrete Choice Experiments**.

Author(s)

Hideo Aizaki

References

Burgess, L. (2007) *Discrete Choice Experiments [computer software]*, Department of Mathematical Sciences, University of Technology, Sydney, available at http://maths.science.uts.edu.au/maths/wiki/SPExpts

Street, D. J., and Burgess, L. (2007) *The Construction of Optimal Stated Choice Experiments*. Hoboken, NJ: John Wiley & Sons.

See Also

oa.design, Lma.design, rotation.design

Examples

```
# The following lines of code reproduce five examples shown on
# the website Discrete Choice Experiments.

# Check Main Ex. 1
## Not run:
CheckMainEx1.sets <- matrix(c(
  0, 0, 0, 0, 3, 2, 2, 2,
  2, 2, 0, 2, 1, 1, 2, 1,
  3, 0, 0, 0, 2, 2, 2, 2,
  1, 1, 0, 1, 0, 0, 2, 0,
  1, 0, 2, 0, 0, 2, 1, 2,
  3, 2, 1, 0, 2, 1, 0, 2,
  2, 1, 0, 0, 1, 0, 2, 2,
  0, 0, 0, 2, 3, 2, 2, 1,
  3, 0, 0, 1, 2, 2, 2, 0,
  0, 2, 2, 1, 3, 1, 1, 0,
```

```
  1, 2, 0, 0, 0, 1, 2, 2,
  1, 0, 1, 2, 0, 2, 0, 1,
  2, 0, 2, 0, 1, 2, 1, 2,
  2, 0, 1, 1, 1, 2, 0, 0,
  3, 1, 2, 2, 2, 0, 1, 1,
  0, 1, 1, 0, 3, 0, 0, 2),
  nrow = 16, byrow = TRUE)

CheckMainEx1 <- sb.design(
  operation       = "check",
  nattributes     = 4,
  nlevels         = c(4, 3, 3, 3),
  nalternatives   = 2,
  attribute.names = list(A = c("a0", "a1", "a2", "a3"),
                         B = c("b0", "b1", "b2"),
                         C = c("c0", "c1", "c2"),
                         D = c("d0", "d1", "d2")),
  design          = CheckMainEx1.sets,
  effect          = "main")

CheckMainEx1
summary(CheckMainEx1)
## End(Not run)

# Construct Main Ex. 1
## Not run:
ConstructMainEx1.treatment <- matrix(c(
  0, 0, 0, 0,
  2, 2, 0, 2,
  3, 0, 0, 0,
  1, 1, 0, 1,
  1, 0, 2, 0,
  3, 2, 1, 0,
  2, 1, 0, 0,
  0, 0, 0, 2,
  3, 0, 0, 1,
  0, 2, 2, 1,
  1, 2, 0, 0,
  1, 0, 1, 2,
  2, 0, 2, 0,
  2, 0, 1, 1,
  3, 1, 2, 2,
  0, 1, 1, 0),
  nrow = 16, byrow = TRUE)

ConstructMainEx1 <- sb.design(
  operation       = "construct",
  nattributes     = 4,
  nlevels         = c(4, 3, 3, 3),
  nalternatives   = 2,
  attribute.names = list(A = c("a0", "a1", "a2", "a3"),
                         B = c("b0", "b1", "b2"),
                         C = c("c0", "c1", "c2"),
```

```
                          D = c("d0", "d1", "d2")),
  design          = ConstructMainEx1.treatment,
  generators      = c(3, 2, 2, 2),
  effect          = "main")

ConstructMainEx1
summary(ConstructMainEx1)
## End(Not run)

# Construct Main Ex. 2
## Not run:
ConstructMainEx2.treatment <- matrix(c(
  0, 0,
  0, 1,
  1, 0,
  1, 1),
  nrow = 4, byrow = TRUE)

ConstructMainEx2 <- sb.design(
  operation       = "construct",
  nattributes     = 2,
  nlevels         = c(2, 2),
  nalternatives   = 2,
  attribute.names = list(X = c("x0", "x1"),
                         Y = c("y0", "y1")),
  design          = ConstructMainEx2.treatment,
  generators      = c(1, 1),
  effect          = "main")

ConstructMainEx2
summary(ConstructMainEx2)
## End(Not run)

# Construct Main + Some Ex. 1
## Not run:
ConstructMainSomeEx1.treatment <- matrix(c(
  0, 0, 0, 0,
  2, 2, 0, 2,
  3, 0, 0, 0,
  1, 1, 0, 1,
  1, 0, 2, 0,
  3, 2, 1, 0,
  2, 1, 0, 0,
  0, 0, 0, 2,
  3, 0, 0, 1,
  0, 2, 2, 1,
  1, 2, 0, 0,
  1, 0, 1, 2,
  2, 0, 2, 0,
  2, 0, 1, 1,
  3, 1, 2, 2,
  0, 1, 1, 0),
  nrow = 16, byrow = TRUE)
```

```
ConstructMainSomeEx1 <- sb.design(
  operation       = "construct",
  nattributes     = 4,
  nlevels         = c(4, 3, 3, 3),
  nalternatives   = 2,
  attribute.names = list(A = c("a0", "a1", "a2", "a3"),
                         B = c("b0", "b1", "b2"),
                         C = c("c0", "c1", "c2"),
                         D = c("d0", "d1", "d2")),
  design          = ConstructMainSomeEx1.treatment,
  generators      = c(3, 2, 2, 2),
  effect          = "mplussome",
  interactions    = c(1, 2),
  determinant     = "1")

ConstructMainSomeEx1
summary(ConstructMainSomeEx1)
## End(Not run)

# Construct Main + Some Ex. 2
## Not run:
ConstructMainSomeEx2.treatment <- matrix(c(
  0, 0,
  0, 1,
  1, 0,
  1, 1),
  nrow = 4, byrow = TRUE)

ConstructMainSomeEx2 <- sb.design(
  operation       = "construct",
  nattributes     = 2,
  nlevels         = c(2, 2),
  nalternatives   = 2,
  attribute.names = list(X = c("x0", "x1"),
                         Y = c("y0", "y1")),
  design          = ConstructMainSomeEx2.treatment,
  generators      = matrix(c(0, 1, 1, 1), nrow = 2, byrow = TRUE),
  effect          = "mplussome",
  interactions    = c(1, 2),
  determinant     = NULL)

ConstructMainSomeEx2
summary(ConstructMainSomeEx2)
## End(Not run)
```

---

syn.res1, syn.res2, syn.res3

Synthetic respondent data sets

Description

Data sets artificially created for the "Examples" in this package.

Usage

```
data(syn.res1)
data(syn.res2)
data(syn.res3)
```

Format

Data frames with 100 respondents on the following 11 or 12 variables.

`ID` Identification number of respondents.

`BLOCK` Serial number of blocks to which each respondent had been assigned.

`q1` Response to choice experiment question 1.

`q2` Response to choice experiment question 2.

`q3` Response to choice experiment question 3.

`q4` Response to choice experiment question 4.

`q5` Response to choice experiment question 5.

`q6` Response to choice experiment question 6.

`q7` Response to choice experiment question 7.

`q8` Response to choice experiment question 8.

`q9` Response to choice experiment question 9.

`F` Female dummy variable (`1` = Female, otherwise `0`) that is included only in `syn.res1`.

Details

The "Examples" for each function in this package is based on three hypothetical cases in which choice experiments are applied to measure consumers' valuation of an agricultural product.

The agricultural product has three attributes: (1) the region of origin: this attribute has three levels—"Region A," "Region B," and "Region C."; (2) the eco-friendly label: this describe the three types of cultivation method—"Conv. (conventional cultivation method)," "More (more eco-friendly cultivation method)," and "Most (most eco-friendly cultivation method); and (3) the price per piece of the product—"$1," "$1.1," "$1.2."

- `syn.res1` (Unlabeled choice experiments)
  The data set `syn.res1` is based on a case in which an unlabeled choice experiment design created by the mix-and-match method is used in a questionnaire survey. A total of 100 respondents were assumed to have been requested to select their most preferred from among two agricultural products and the option "none of these."

  Q1. Please select your most preferred alternative from the following:

  Product 1 Product 2

Region of origin	Region B	Region C
Eco-friendly	More	Most
Price	$1.1	$1.2

1. I select product 1.
2. I select product 2.
3. I select none of these.

A total of 9 choice experiment questions are created by the function `rotation.design` . Each respondent had to respond to a total of 9 choice experiment questions, implying that the sample size of the analysis based on their responses was 900 (= 9 choice experiment questions per respondent * 100 respondents). In the example, the effect of the respondents' gender on their valuations of the cultivation methods was also examined (see "Examples" for the function `make.dataset`). See the first case in "Example" for the function `make.dataset`.

- `syn.res2` (Labeled choice experiments)

  The data set `syn.res2` is based on a case in which a labeled choice experiment design created by the *L^MA* method is used in a questionnaire survey. A total of 100 respondents were assumed to have been requested to select their most preferred from among three agricultural products and the option "none of these."

  Q1. Please select your most preferred alternative from the following:

	Region A	Region B	Region C
Eco-friendly	Most	Conv.	More
Price	$1.1	$1	$1.1

1. I select the Region A product.
2. I select the Region B product.
3. I select the Region C product.
4. I select none of these.

  Although the agricultural products have also three attributes and their levels mentioned above, the region of origin attribute is treated as an alternative specific attribute: the first, second, and third alternatives in a choice set always read as "Region A," "Region B," and "Region C," respectively.

  A total of 18 choice experiment questions were created by the function `Lma.design` and divided into two blocks: this means that two types of questionnaire were created, each of which was randomly assigned to the respondents. Therefore, the sample size of the analysis based on their responses was 900 (= 9 choice experiment questions per respondent * 100 respondents). See the second case in "Examples" for the function `make.dataset`.

- `syn.res3` (Binary choice experiments with an opt-out option)

  The data set `syn.res3` is based on a case in which a binary choice experiment design created by the *L^MA* method is used in a questionnaire survey. A total of 100 respondents were assumed to have been requested to decide whether they select an agricultural product or not.

Q1. Would you like to purchase the following?

	Product
Region of origin	Region A
Eco-friendly	Conv.
Price	$1

1. Yes.
2. No.

A total of 9 binary choice experiment questions are created by the function `Lma.design`. Each respondent had to respond to a total of 9 binary choice experiment questions, implying that the sample size of the analysis based on their responses was 900 (= 9 binary choice experiment questions per respondent * 100 respondents). See the last case in "Examples" for the function `make.dataset`.

Author(s)

Hideo Aizaki

References

Aizaki, H. (2012) Basic Functions for Supporting an Implementation of Choice Experiments in R. *Journal of Statistical Software, Code Snippets*, **50**(2), 1–24. http://www.jstatsoft.org/v50/c02/

See Also

`make.dataset`, `rotation.design`, `Lma.design`

Examples

```
# See "Examples" for the function make.dataset.
```

Index