Journal of Statistical Software

September 2012, Volume 50, Code Snippet 2. *http://www.jstatsoft.org/*

Basic Functions for Supporting an Implementation of Choice Experiments in R

Hideo Aizaki
National Agriculture and Food Research Organization

Abstract

The package **support.CEs** provides seven basic functions that support the implementation of choice experiments (CEs) in R: two functions for creating a CE design, which is based on orthogonal main-effect arrays; a function for converting a CE design into questionnaire format; a function for converting a CE design into a design matrix; a function for making the data set suitable for the implementation of a conditional logit model; a function for calculating the goodness-of-fit measures of an estimated model; and a function for calculating the marginal willingness to pay for the attributes and/or levels of the estimated model.

Keywords: choice set, conditional logit model, marginal willingness to pay, questionnaire, rho-squared, **survival**, **DoE.base**.

1. Introduction

Choice experiments (CEs) – also known as discrete choice experiments, stated (discrete) choice methods, attribute-based stated preference methods, or choice-based conjoint analyses – have became one of the most important statistical methods used by studies across various research areas in the social sciences (Bateman *et al.* 2002; Bennett and Blamey 2001; Hensher, Rose, and Greene 2005; Holmes and Adamowicz 2003; Kanninen 2007; Louviere, Hensher, and Swait 2000; Ryan, Gerard, and Amaya-Amaya 2008). Although many of these studies have outlined various ways that CEs can be applied, it may not be easy for those who are relatively unfamiliar with CEs to create a CE design and conduct statistical analyses of responses to CE questions without additional support.

The package **support.CEs** (Aizaki 2012) provides seven basic functions for CEs in R (R Development Core Team 2012): two functions for creating a CE design based on orthogonal main-effect arrays; a function for converting a CE design into questionnaire format; a function

for converting a CE design into a design matrix; a function for making the data set suitable for implementing a conditional logit (CL) model; a function for calculating the goodness-of-fit measures of an estimated model; and a function for calculating the marginal willingness to pay (MWTP) for the attributes and/or levels of the estimated model. Some of these functions are developed on the basis of Aizaki and Nishimura's note on conducting CEs in R (Aizaki and Nishimura 2008) and a macro program, which works in the Japanese environment, creates a CE design, and conducts CE analyses based on Microsoft **Excel** connected with R (Aizaki 2009).

2. Outline of choice experiments and the package support.CEs

2.1. Steps in choice experiments

CEs can be applied to situations wherein an individual selects one alternative from a set of alternatives – each of which is expressed by a bundle of attributes – which reveals important attributes/levels affecting this individual's selection. Such situations are widely observed in our daily lives – selecting a travel mode, purchasing food in a market, selecting a clinic, voting in an election, and so on – and therefore, CEs have been applied in many disciplines.

Before outlining a method for implementing CEs, the terms in this paper are defined. 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 CEs 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 CE, respondents are usually asked to select the most preferred alternative from a choice set; therefore, one choice set constitutes a CE question. A "CE design" refers to a collection of individual choice sets.

Figure 1 shows an example of a CE design. The CE design includes a total of 9 choice set (Q1 to Q9). Each choice set (question) consists of three alternatives ("Alternative 1," "Alternative 2," and "none of these" option). "Altenative 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."

After establishing the objectives of applying CEs, CE implementation can roughly be summarized in three steps: (1) examining a hypothetical situation and creating a CE design; (2) preparing and conducting a questionnaire survey; and (3) preparing a data set and conducting an analysis (for details, see Bateman *et al.* 2002, Bennett and Blamey 2001, Hensher *et al.* 2005, Holmes and Adamowicz 2003, Kanninen 2007, Louviere *et al.* 2000, Ryan *et al.* 2008).

In the first step, the CE user has to consider a hypothetical situation wherein respondents are requested to answer CE questions using one or more of the following methods: reviewing previous research, interviewing experts/people involved with the issue of decision making in CE questions, conducting fieldwork, and conducting pilot surveys and/or focus groups. After constructing a hypothetical situation, a CE design is accordingly decided upon and created. There are roughly two approaches: a manual/catalog-based design approach that creates a CE

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

1. I select alternative 1.
2. I select alternative 2.
3. 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

1. I select alternative 1.
2. I select alternative 2.
3. 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

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

Figure 1: An example of a CE design.

design using an orthogonal array, and a computational search-based optimal design approach that creates a CE design from the viewpoint of design efficiency measures (e.g., Johnson, Kanninen, Bingham, and Özdemir 2007).

The second step in implementing a CE is to prepare and conduct a questionnaire survey. Although this step is similar to that employed in general questionnaire surveys (e.g., Mangione 1995; Dillman 2000), it should be noted that the sample size is different from the number of respondents. Each respondent is asked multiple CE questions and the sample size is calculated by multiplying the number of respondents by the number of questions per respondent (see Hensher *et al.* 2005 for a method that considers the number of respondents in CE applications).

In the final step in implementing a CE, the CE user has to prepare a data set from valid samples and conduct a statistical analysis on the basis of the data set by using discrete choice models. A popular discrete choice model for CE applications is a CL model, which is

slightly different from a multinomial logit model, although the model names are frequently confused (see Greene 2011 or Wooldridge 2010 for the differences between the two models). The probability of individual n selecting alternative i from choice set C_n in the CL model is as follows:

$$P(i) = \exp(V_{in}) / \sum_{j \in C_n} \exp(V_{jn}).$$

V_{in} is a systematic component of utility that is usually assumed to be a linear additive function of the independent variables X_{ikn} with coefficients β_{ik}, as follows:

$$V_{in} = \sum_{k=1}^{K} \beta_{ik} X_{ikn}.$$

It is necessary to calculate goodness-of-fit measures to evaluate an estimated CL model. A well-known measure is ρ^2 (rho-squared) – also called McFadden's R^2 or pseudo R^2 – which can be defined as follows (Ben-Akiva and Lerman 1985):

$$\rho^2 = 1 - (LL_b / LL_0),$$

where LL_b and LL_0 are the log likelihood values at convergence and at the start, respectively. The measure adjusted by the number of coefficients (K) is defined as follows:

$$\bar{\rho}^2 = 1 - (LL_b - K / LL_0).$$

Although a lot of useful information can be derived from an estimated model, the MWTP for attributes/levels is widely used in researches in which CEs are applied. In the CL model – which has a simple linear utility function – an MWTP for a non-monetary variable is calculated as $-b_{nm}/b_m$; where, b_{nm} is an estimated coefficient of the non-monetary variable, and b_m is an estimated coefficient of the monetary variable. Further, confidence intervals for the MWTPs are frequently calculated using a simulation method on the basis of a vector of means and the variance-covariance matrix of estimates proposed by Krinsky and Robb (1986).

2.2. Roles of the package

Table 1 shows a classification of the functions in package **support.CEs** according to the steps in CEs. In the first step, there are two functions – `rotation.design` and `Lma.design` – for creating a CE design according to the catalog-based approach explained by Johnson *et al.* (2007) – see also Chrzan and Orme 2000 and associated references. The function `rotation.design` executes a rotation design method and a mix-and-match design method. The rotation method uses the orthogonal main-effect array as the first alternative in each choice set; this method

Steps	Functions
Examining a hypothetical situation and creating a CE design	`rotation.design`, `Lma.design`
Preparing and conducting a questionnaire survey	`questionnaire`
Preparing a data set and conducting a statistical analysis	`make.design.matrix`, `make.dataset`, `gofm`, `mwtp`

Table 1: Functions in each step of the CEs.

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 is/are created using the rotation method and placed into different urn(s). A choice set is generated by randomly selecting one alternative from each urn. This selection process is repeated, without replacement, until all the alternatives are assigned to N choice sets. These N choice sets correspond to a CE design.

The function `Lma.design` realizes an L^{MA} design method that directly creates a CE design from the orthogonal main-effect array. In this L^{MA} design 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.

The functions `rotation.design` and `Lma.design` have the following three features. (1) While the rotation and mix-and-match methods create an unlabeled type of CE design that can contain generic attributes in the utility function of the CL model, the L^{MA} method creates a labeled type of CE 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. (2) Each method depends on an orthogonal main-effect array that is generated using the function `oa.design` in the package **DoE.base** (Grömping 2012). (3) Each function provides an option by which the CE design can be divided into sub-blocks.

The function `questionnaire` is developed by way of preparing and conducting the questionnaire survey. This function converts a CE design (created by the function `rotation.design` or `Lma.design`) into CE questions that can be used in a questionnaire survey. The function `questionnaire` decreases the overall time spent in creating a questionnaire.

CL model analyses of responses to CE questions in R can be conducted using the function `clogit` in the package **survival** (Therneau 2012). When this function is used to analyze the responses to the CE questions, a data set in a special format is needed; each alternative should comprise one row of this data set. In order to support the creation of a data set suitable for the function `clogit`, two functions have been prepared. The first is the function `make.design.matrix`: this function converts a CE design (created by the function `rotation.design` or `Lma.design`) into a design matrix, where each alternative comprises one row of the matrix. The second is the function `make.dataset`: this function creates a data set that can be used for the function `clogit` by combining a data set containing information about responses to the CE questions (a structure where one row shows one respondent) and a data set containing a design matrix related to these questions (created by the function `make.design.matrix`).

Although the function `clogit` returns a significant amount of information about an estimated model, ρ^2 cannot be calculated. Therefore, the function `gofm` that provides ρ^2 and $\bar{\rho}^2$ is developed. This function also displays the number of estimated coefficients, and log likelihood values that are automatically calculated in the function `clogit` at the start and

at convergence. In order to calculate MWTP from the CL model estimates, the function `mwtp` is developed. The function `mwtp` provides MWTP in the case of the simple linear utility function and confidence interval for the MWTP according to the simulation method (Krinsky and Robb 1986).

3. Usage of functions

3.1. Creating a choice experiment design

Two design functions – the function `rotation.design`, corresponding to the rotation design method and the mix-and-match design method, and the function `Lma.design`, corresponding to the L^{MA} design method – have six arguments in common (Table 2). The combination of attributes and attribute levels are assigned to the argument `attribute.names` in list format. For example, let us 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 `attribute.names` is set to be `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 alternative, such as the option "none of these." For example, in a choice set, if there are two product alternatives and the option "none

Argument	Description
`candidate.array`	A data frame containing an array created by the user. Normally, when these functions are used, this argument does not need to be set by the user. Therefore, the default value is `NULL`.
`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."
`nblocks`	An integer value describing the number of blocks into which a CE design is divided.
`row.renames`	A logical variable describing whether or not the row names of a CE design created by these functions 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 orthogonal main-effect array created by the function `oa.design` (included in the package **DoE.base**), 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. This arugument is only valid for the function `rotation.design`.
`seed`	Seed for a random number generator.

Table 2: Arguments for the functions `rotation.design` and `Lma.design`.

of these" (see Figure 1), then the argument `nalternatives` is set to be `2`.

When a large CE design is created (that is, a CE that has numerous questions), the respondent may carry a heavy psychological burden in terms of answering the questions; in these cases, the CE 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 this argument is set to be 3 and the CE design contains 27 individual choice sets (that is, there are 27 CE questions), the CE design is divided into 3 blocks, each of which has 9 individual choice sets (9 CE questions). In the function `Lma.design`, blocking is performed on the basis of a factor with `nblocks` levels; however, the function `rotation.design` randomly divides a CE design into sub-blocks based on the argument `nblocks`. Therefore, in the case of the function `rotation.design`, the argument `nblocks` must be divisors of the number of choice sets included in the CE design.

In order to create a CE design under default settings, the two functions use an orthogonal main-effect array that is automatically produced by the function `oa.design` based on the argument `attribute.names`. However, when there is no orthogonal main-effect array corresponding to the argument `attribute.names`, the function `oa.design` returns a full factorial based on the argument `attribute.names`. When the two functions do not create a CE design matching the user's requirements, the user might achieve it by assigning an arbitrary (user-defined) array to the argument `candidate.array`: the functions then use the array to create a CE 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.

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

Each design function returns three types of outputs in list format. The first is a CE design that is provided as a list (`alternatives`) of objects, `alt.j` ($j = 1, 2, ..., J$), where each `alt.j` includes the j-th alternative in each choice set created by the functions. Each `alt.j` object contains attribute variables corresponding to the argument `attribute.names`, a variable `BLOCK` describing the serial number of blocks, a variable `QES` describing the serial number of CE questions for each value of the variable `BLOCK`, and a variable `ALT` describing the serial number of alternatives for each value of the variable `QES`. The second type of output is the `candidate` object: this object contains the array used for the candidate set, 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 the candidate set shows a factor that is used for blocking. The third type of output (`design.information`) is information related to the CE design generated by the design functions, 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 CE design is divided (`nblocks`), the number of questions per block (`nquestions`), the number of alternatives per choice set excluding an opt-out alternative (`nalternatives`), and the number of attributes per alternative (`nattributes`).

In the case of the function `rotation.design`, an error message is displayed when the argument `nblocks` does not match a CE design. In such a case, the argument `nblocks` is set to be a divisor of the number of rows of the CE design. Other messages are frequently shown

Argument	Description
`choice.experiment.design`	A CE design created by the function `rotation.design` or `Lma.design`.
`categorical.attributes`	A vector containing the names of attributes treated as categorical independent variables in the CL model analysis.
`continuous.attributes`	A vector containing the names of attributes treated as continuous independent variables in the CL model analysis.
`unlabeled`	A logical variable describing the types of CE designs 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`.
`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.

Table 3: Arguments for the function `make.design.matrix`.

immediately after executing two functions when these functions work 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`.

3.2. Converting a choice experiment design into questionnaire format

The function `questionnaire` converts a CE design created by the function `rotation.design` or `Lma.design` into CE questions used in a questionnaire survey. The CE design is assigned to the argument `choice.experiment.design`. CE questions converted from the CE design are returned in order of the values of the number of blocks and number of questions.

3.3. Converting a choice experiment design into a design matrix

The function `make.design.matrix` is able to convert a CE design created by the function `rotation.design` or `Lma.design` into a design matrix that is suitable for the CL model analysis with the function `clogit` in the package **survival**. There are five arguments defined in the function `make.design.matrix` (Table 3). The argument `choice.experiment.design` is assigned a CE design created by the function `rotation.design` or `Lma.design`.

Attributes included in the CE 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 – X, Y, and Z. In the CL model analysis, when the attributes X and Y are treated as categorical variables and the attribute Z is treated as a continuous variable, the argument `categorical.attributes` is set as `c("X", "Y")`, and the argument `continuous.attributes` is set as `c("Z")`.

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 of 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, below). 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 – "x1," "x2," and "x3" – dummy variables are created as follows: (1) When the CE design is unlabeled, two dummy variables are created: a dummy variable x2 that assumes a value of 1 when the attribute X takes "x2," and 0 otherwise; and a dummy variable x3 that assumes a value of 1 when the attribute X takes "x3," and 0 otherwise. (2) When the CE 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 the attribute X in alternative 1 takes "x2," and 0 otherwise; a dummy variable x22 that assumes a value of 1 when the attribute X in alternative 2 takes "x2," and 0 otherwise; a dummy variable x31 that assumes a value of 1 when the attribute X in alternative 1 takes "x3," and 0 otherwise; and a dummy variable x32 that assumes a value of 1 when the attribute X in alternative 2 takes "x3," and 0 otherwise.

The argument unlabeled is set as TRUE when the CE design assigned to the argument choice.experiment.design is unlabeled; it is FALSE otherwise (when the CE 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 CE questions. It is set as FALSE when the opt-out alternative is not included.

The function make.design.matrix provides a design matrix that can be directly assigned to the argument design.matrix in the function make.dataset (Figure 2). The design matrix contains categorical and/or continuous variables created by the function make.design.matrix as well as the following four kinds of variables: BLOCK, QES, ALT, and ASC. BLOCK is an integer variable describing the serial number of the blocks; QES is an integer serial number of the questions, which starts at 1 in each block; ALT is an integer variable describing the serial number of alternatives according to the value of the variable QES; ASC refers to alternative specific constant(s). When the CE design is labeled, the serial number of the alternatives is automatically appended to the tail of ASC (such as ASC1, ASC2, and ASC3). In the example shown in Figure 2, attributes are assumed to be the categorical attributes X and Y, and the continuous attribute Z, a CE design is supposed to be divided into three blocks (BLOCK values are 1, 2, and 3), and each question is assumed to have three alternatives including a "none of these" option. The structure of the design matrix is such that each row shows one alternative: the first row shows the first alternative (ALT = 1) in the first question (QES = 1) in the first block (BLOCK = 1); the second row shows the second alternative (ALT = 2) in the first question (QES = 1) in the first block (BLOCK = 1); and the last row shows the third alternative (ALT = 3) in the third question (QES = 3) in the third block (BLOCK = 3).

	BLOCK	QES	ALT	ASC	x2	x3	y2	y3	z
1	1	1	1	1	0	1	1	0	10
2	1	1	2	1	0	0	0	1	20
3	1	1	3	0	0	0	0	0	0
4	1	2	1	1	0	0	0	0	10
5	1	2	2	1	0	0	1	0	30
6	1	2	3	0	0	0	0	0	0
7	1	3	1	1	1	0	0	1	10
8	1	3	2	1	0	0	0	0	10
9	1	3	3	0	0	0	0	0	0
10	2	1	1	1	0	1	0	0	20
...									
25	3	3	1	1	0	0	1	0	30
26	3	3	2	1	1	0	0	0	30
27	3	3	3	0	0	0	0	0	0

Figure 2: An example of a design matrix data set.

3.4. Making a data set

The function `make.dataset` is able to create a data set suitable for the function `clogit` by combining a data set containing information about responses to the CE questions and a data set containing a design matrix related to these questions.

There are four arguments in the function `make.dataset` (Table 4). 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 the variable `ID`, corresponding to the respondent's identification number; the variable `BLOCK`, corresponding to the serial number of blocks to which each respondent is assigned; and response variables, corresponding to the answers to each of the CE questions. If necessary, covariates showing the respondent's individual characteristics such as age and gender are also included in the respondent data set. 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,

Argument	Description
`respondent.dataset`	The name of the data frame containing the respondents' answers to the CE questions.
`design.matrix`	The name of the data frame containing a design matrix created by the function `make.design.matrix` in this package.
`choice.indicators`	A vector of the names of variables showing the alternative of which was selected in each CE 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. Its default is `NULL`.

Table 4: Arguments for the function `make.dataset`.

	ID	BLOCK	q1	q2	q3
1	1	1	1	3	1
2	2	2	2	1	1
3	3	3	1	2	1
4	4	1	1	1	1
5	5	2	2	2	1
6	6	3	1	2	1
7	7	1	2	1	1
8	8	2	2	2	1
9	9	3	2	2	3
10	10	1	1	1	1
...					
97	97	1	2	1	1
98	98	2	2	1	1
99	99	3	1	2	2

Figure 3: An example of a respondent data set.

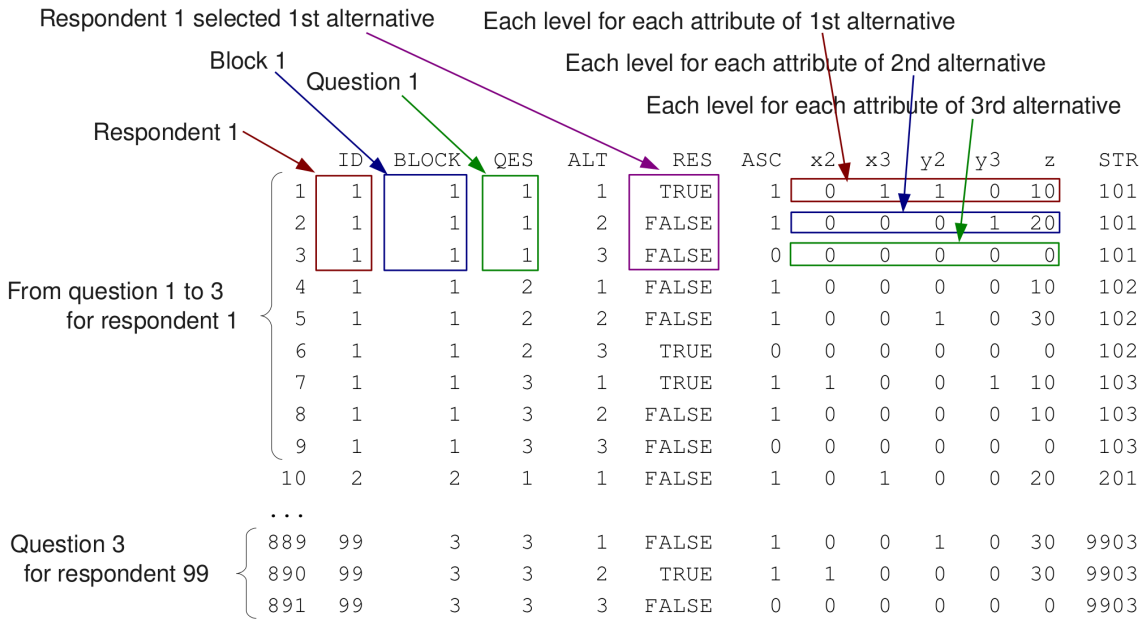
and q3, the argument is set as `c("q1", "q2", "q3")`. The response variables show the serial number of the alternative selected by the respondent for each CE question. The method of assigning the serial number of the alternatives must be the same as that of assigning `ALT` in the output from the function `make.design.matrix`. In other words, each alternative must be assigned an integer value that starts from 1. In `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`.

For example, Figure 3 shows an artificial respondent data set that contains 99 respondents' answers to CE questions. It is assumed that each respondent answers a questionnaire containing an arbitrarily assigned block (any one of three) of a CE design; each respondent is asked three CE questions, each of which has three alternatives, and the responses are stored in the `q1`, `q2`, and `q3` variables. The answers to the CE questions can be read as follows: for example, the first row shows that respondent 1 (`ID = 1`) was asked to respond to a questionnaire containing the first block (`BLOCK = 1`) of the CE design and selected the first (`1`), third (`3`), and first (`1`) alternatives for the first (`q1`), second (`q2`), and third (`q3`) CE questions, respectively. The last row shows that respondent 99 (`ID = 99`) was asked to respond to a questionnaire containing the third block (`BLOCK = 3`) of the CE design and selected the first (`1`), second (`2`), and second (`2`) alternatives for the first (`q1`), second (`q2`), and third (`q3`) CE questions, respectively.

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 data set 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 CE questions presented to respondents was randomized.

The function `make.dataset` returns the data set used for the function `clogit` (Figure 4). In addition to some variables contained in the respondent and design matrix data sets, this data set also contains the following two variables that are used in the function `clogit`: the first variable is `STR`, which is assigned to the argument `strata` in the function `clogit` in order to identify each combination of respondent and question; the second variable is `RES`, which corresponds to a logical variable that takes on `TRUE` when the alternative is selected and `FALSE`



	ID	BLOCK	QES	ALT	RES	ASC	x2	x3	y2	y3	z	STR
1	1	1	1	1	TRUE	1	0	1	1	0	10	101
2	1	1	1	2	FALSE	1	0	0	0	1	20	101
3	1	1	1	3	FALSE	0	0	0	0	0	0	101
4	1	1	2	1	FALSE	1	0	0	0	0	10	102
5	1	1	2	2	FALSE	1	0	0	1	0	30	102
6	1	1	2	3	TRUE	0	0	0	0	0	0	102
7	1	1	3	1	TRUE	1	1	0	0	1	10	103
8	1	1	3	2	FALSE	1	0	0	0	0	10	103
9	1	1	3	3	FALSE	0	0	0	0	0	0	103
10	2	2	1	1	FALSE	1	0	1	0	0	20	201
...												
889	99	3	3	1	FALSE	1	0	0	1	0	30	9903
890	99	3	3	2	TRUE	1	1	0	0	0	30	9903
891	99	3	3	3	FALSE	0	0	0	0	0	0	9903

Figure 4: An example of a data set suitable for the function `clogit`.

when it is not. An example data set (shown in Figure 4) is created from the design matrix of Figure 2 and the respondent data set of Figure 3. Since each row shows one alternative and each respondent is requested to answer three CE questions, a set of 9 rows shows one respondent's data set: the first to the ninth rows show respondent 1's (`ID = 1`) responses to three CE questions, each of which has three alternatives. From the first three rows, we can read that respondent 1 was made to respond to the first block of the CE design (`BLOCK = 1`) and selected the first alternative (`RES` in row 1 `= TRUE`, and `RES`s in rows 2 and 3 `= FALSE`) for the first CE question (`QES = 1`). Similarly, the last three rows (rows 889–891) show that respondent 99 (`ID = 99`) was made to respond to the third block of the CE design (`BLOCK = 3`) and selected the second alternative (`RES` in row 890 `= TRUE`, and `RES`s in rows 889 and 891 `= FALSE`) for the third CE question (`QES = 3`).

3.5. Calculating goodness-of-fit measures

The function `gofm` has an argument `output`, which refers to the name of an object containing the output from the function `clogit`, and provides the goodness-of-fit measures for the CL model: ρ^2 (`RHO2`) and ρ^2 adjusted by the number of estimated coefficients (`AdjRHO2`). The function `gofm` also displays the number of estimated coefficients (`K`) and the log likelihood values that are automatically calculated in the function `clogit` at the start (`LL0`) and at convergence (`LLb`).

3.6. Calculating MWTPs

The function `mwtp` calculates the MWTPs for attributes/levels and confidence intervals of the MWTPs according to the method proposed by Krinsky and Robb (1986). The definition of the MWTP of a non-monetary variable provided by the function `mwtp` is $-b_{nm}/b_m$; where, b_{nm}

Argument	Description
`output`	The name of an object containing the output from the function `clogit` in the package **survival**.
`monetary.variables`	A vector containing the names of the monetary variables in the output used to calculate the MWTPs.
`nonmonetary.variables`	A vector or list of vectors 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`	A vector of values showing the percentile points of an empirical distribution of each MWTP. Its default vector is set as `c(0.5, 2.5, 5, 95, 97.5, 99.5)`, which indicates the lower and upper bounds of the 99, 95, and 90 percent confidence intervals.
`seed`	Seed for a random number generator.

Table 5: Arguments for the function `mwtp`.

is the estimated coefficient of the non-monetary variable, and b_m is the estimated coefficient of a monetary variable.

There are six arguments in the abovementioned function (Table 5). 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 CE 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.

4. Examples

Two hypothetical examples of how all functions in the package **support.CEs** can be used are given on the basis of a situation wherein consumers are assumed to provide valuations of an agricultural product. The 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 describes the three types of cultivation methods – "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: this attribute has three levels – "$1," "$1.1," and "$1.2." A total of 100 respondents were assumed to have been requested to select their most preferred from a set of products; they were also allowed the option "none of these." The objective of applying CEs in the examples was to measure the respondents' relative importance of the region of origin and the cultivation methods through the application of CEs. Responses to CE questions in the examples were artificially generated using random numbers.

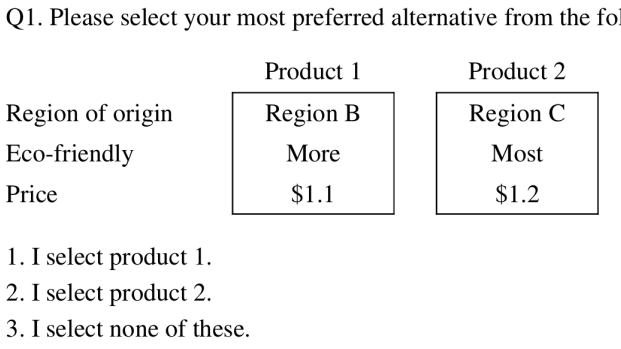


Figure 5: A CE question in the unlabeled design example.

4.1. Example of an unlabeled design

The first example is a case in which an unlabeled CE design created by the mix-and-match method is used in a questionnaire survey. Respondents were assumed to have been requested to select their most preferred from among two agricultural products and the option "none of these" (Figure 5). In the example, the effect of the respondents' gender on their valuations of the cultivation methods was also examined. The function `rotation.design` with the arguments assigned according to the abovementioned condition was executed as follows:

```
R> 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)

The columns of the array have been used in order of appearance.
For designs with relatively few columns,
the properties can sometimes be substantially improved
using option columns with min3 or even min34.

R> des1

Choice sets:
alternative 1 in each choice set
  BLOCK QES ALT Region   Eco Price
1     1   1   1  Reg_B  More   1.1
8     1   2   1  Reg_B  Most     1
9     1   3   1  Reg_A  Most   1.1
6     1   4   1  Reg_A Conv.     1
3     1   5   1  Reg_C  Most   1.2
7     1   6   1  Reg_A  More   1.2
5     1   7   1  Reg_C Conv.   1.1
4     1   8   1  Reg_B Conv.   1.2
2     1   9   1  Reg_C  More     1
```

```
alternative 2 in each choice set
  BLOCK QES ALT Region   Eco Price
1     1   1   2  Reg_C  Most   1.2
5     1   2   2  Reg_A  More   1.2
8     1   3   2  Reg_C Conv.   1.1
4     1   4   2  Reg_C  More     1
3     1   5   2  Reg_A Conv.     1
9     1   6   2  Reg_B Conv.   1.2
2     1   7   2  Reg_A  Most   1.1
7     1   8   2  Reg_B  Most     1
6     1   9   2  Reg_B  More   1.1

Candidate design:
  A B C
1 2 2 2
2 3 2 1
3 3 3 3
4 2 1 3
5 3 1 2
6 1 1 1
7 1 2 3
8 2 3 1
9 1 3 2
class=design, type= oa

Design information:
number of blocks = 1
number of questions per block = 9
number of alternatives per choice set = 2
number of attributes per alternative = 3
```

Messages displayed immediately after executing the function `rotation.design` are taken from the function `oa.design`. The size of the above-generated CE design is nine and the number of blocks is one; that is, each respondent had to respond to a total of nine CE questions, implying that the sample size of the analysis based on their responses was 900 (= 9 CE questions per respondent × 100 respondents).

After creating the CE design, the function `questionnaire` with the argument on the basis of the CE design was executed as follows (output is partially omitted):

```
R> questionnaire(choice.experiment.design = des1)

Block 1

Question 1
       alt.1   alt.2
Region "Reg_B" "Reg_C"
Eco    "More"  "Most"
Price  "1.1"   "1.2"
```

```
Question 2
       alt.1   alt.2
Region "Reg_B" "Reg_A"
Eco    "Most"  "More"
Price  "1"     "1.2"

...

Question 8
       alt.1   alt.2
Region "Reg_B" "Reg_B"
Eco    "Conv." "Most"
Price  "1.2"   "1"

Question 9
       alt.1   alt.2
Region "Reg_C" "Reg_B"
Eco    "More"  "More"
Price  "1"     "1.1"
```

Given the above-created CE questions, a questionnaire survey was conducted. In an actual questionnaire survey, the user of the CE would have to create a respondent data set on the basis of the completed questionnaires; however, for the purpose of this experiment, the data set was created and included in the `syn.res1` data set in the package **support.CEs** as follows:

```
R> data("syn.res1")
R> syn.res1[1:3, ]
  ID BLOCK q1 q2 q3 q4 q5 q6 q7 q8 q9 F
1  1     1  2  1  2  2  2  1  2  3  1 0
2  2     1  1  1  1  1  3  1  3  2  2 1
3  3     1  1  2  1  2  2  3  3  3  1 0
```

The `syn.res1` data set contains the variable `ID` that shows the respondent's identification number, the variable `BLOCK` that shows the block number, the response variables (from `q1` to `q9`), and the respondent's gender variable (`F`) that assumes a value of 1 if the respondent is female, and 0 otherwise.

Next, the design matrix, according to the CE design, is created and stored in the `desmat1` object as follows:

```
R> desmat1 <- make.design.matrix(choice.experiment.design = des1,
+   optout = TRUE, categorical.attributes = c("Region", "Eco"),
+   continuous.attributes = c("Price"), unlabeled = TRUE)
R> desmat1[1:3, ]

  BLOCK QES ALT ASC Reg_B Reg_C More Most Price
1     1   1   1   1     1     0    1    0   1.1
2     1   1   2   1     0     1    0    1   1.2
3     1   1   3   0     0     0    0    0   0.0
```

The design matrix data set contains the variables `BLOCK`, `QES`, `ALT`, `ASC`, and other attribute variables, such as `Reg_B`, `Reg_C`, `More`, `Most`, and `Price`.

With the help of the function `make.dataset`, two data sets – `syn.res1` and `desmat1` – are combined to form the `dataset1` data set; this is used for the function `clogit` as follows:

```
R> dataset1 <- make.dataset(respondent.dataset = syn.res1,
+   choice.indicators =
+     c("q1", "q2", "q3", "q4", "q5", "q6", "q7", "q8", "q9"),
+   design.matrix = desmat1)
R> dataset1[1:3, ]

  ID F BLOCK QES ALT   RES ASC Reg_B Reg_C More Most Price STR
1  1 0     1   1   1 FALSE   1     1     0    1    0   1.1 101
2  1 0     1   1   2  TRUE   1     0     1    0    1   1.2 101
3  1 0     1   1   3 FALSE   0     0     0    0    0   0.0 101
```

Before executing `clogit` based on the data set, a systematic component of the utility in the example is identified. By examining the effect of the respondents' gender (the variable F) on their valuation of *More* or *Most*, the systematic component of the utility of respondent n $(= 1, 2, ..., 100)$ for choosing the agricultural product alternative i $(= 1, 2)$ is as follows (the systematic component of utility for the "none of these" option is normalized to zero):

$$\begin{aligned} V_{in} =& ASC_i + b_{RB} Reg\_B_{in} + b_{RC} Reg\_C_{in} + b_{More} More_{in} + b_{Most} Most_{in} + \\ & b_{MoreF} More_{in} F_n + b_{MostF} Most_{in} F_n + b_P Price_{in}, \end{aligned}$$

where ASC refers to an alternative specific constant; Reg_B represents a dummy variable taking a value of 1 if the region of origin attribute is "Region B," and 0 otherwise; Reg_C is a dummy variable taking a value of 1 if the region of origin attribute is "Region C," and 0 otherwise; *More* is a dummy variable taking a value of 1 if the eco-friendly label is "More," and 0 otherwise; *Most* is a dummy variable taking a value of 1 if the eco-friendly label is "Most," and 0 otherwise; F is a dummy variable taking a value of 1 if respondent n is female, and 0 otherwise; and *Price* is the price variable for the price per piece of agricultural product: b_{RB}, b_{RC}, b_{More}, b_{Most}, b_{MoreF}, b_{MostF}, and b_P are coefficients associated with Reg_B, Reg_C, *More*, *Most*, $MoreF$, $MostF$, and *Price*, respectively. The coefficients b_{MoreF} and b_{MostF} show the effect of the respondents' gender on their valuations of the eco-friendly labels, that is, the interactions between F and *More* or *Most*.

After executing `clogit` on the basis of the aforementioned model specification and the data set, the function `gofm` with the argument `output` set as `clogout1` is executed as follows:

```
R> clogout1 <- clogit(RES ~ ASC + Reg_B + Reg_C + More + Most +
+    More:F + Most:F + Price + strata(STR), data = dataset1)
R> clogout1

Call:
clogit(RES ~ ASC + Reg_B + Reg_C + More + Most + More:F + Most:F +
    Price + strata(STR), data = dataset1)
```

```
          coef exp(coef) se(coef)      z       p
ASC     4.6655  1.06e+02    0.733  6.365 2.0e-10
Reg_B  -0.5951  5.51e-01    0.132 -4.520 6.2e-06
Reg_C  -0.3518  7.03e-01    0.114 -3.073 2.1e-03
More    0.5967  1.82e+00    0.166  3.604 3.1e-04
Most    0.7941  2.21e+00    0.149  5.328 9.9e-08
Price  -4.7507  8.65e-03    0.668 -7.111 1.1e-12
More:F -0.0318  9.69e-01    0.196 -0.162 8.7e-01
Most:F  0.0263  1.03e+00    0.173  0.152 8.8e-01

Likelihood ratio test=134  on 8 df, p=0  n= 2700, number of events= 900
```

```
R> gofm(clogout1)

Rho-squared = 0.06753386
Adjusted rho-squared = 0.05944284
Number of coefficients = 8
Log likelihood at start = -988.7511
Log likelihood at convergence = -921.9769
```

The variables `Reg_B` and `Reg_C` have significantly negative coefficients, indicating that respondents evaluated Region B and Region C lower than Region A. Each coefficient of the variables `More` and `Most` is significantly positive, indicating that each of the values of more and most eco-friendly cultivation methods is higher than that of the conventional cultivation method. However, each coefficient of the variables `More:F` and `Most:F` is not significantly different from zero, meaning that the respondents' gender had no significant effect on their valuations of cultivation methods. The coefficient of the variable `Price` is significantly negative, indicating the respondents' preference for a cheaper product.

Finally, the MWTP for each non-monetary variable is estimated by the function `mwtp` as follows:

```
R> mwtp(output = clogout1, monetary.variables = c("Price"),
+    nonmonetary.variables =
+      c("Reg_B", "Reg_C", "More", "Most", "More:F", "Most:F"),
+    percentile.points = c(5, 95), seed = 987)

            MWTP        5%       95%
Reg_B  -0.125275 -0.179652 -0.080924
Reg_C  -0.074060 -0.122809 -0.032739
More    0.125598  0.068697  0.192759
Most    0.167148  0.111027  0.240470
More:F -0.006686 -0.076782  0.062402
Most:F  0.005533 -0.055602  0.066677
```

The output shows that compared to Region A, the MWTPs for Region B and Region C are negative values; compared to the conventional cultivation method, the MWTPs for more and most eco-friendly cultivation method are \$0.126 and \$0.167.

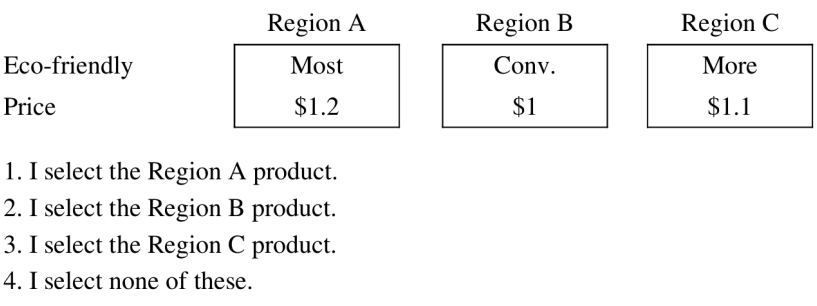


Figure 6: A CE question in the labeled design example.

4.2. Example of a labeled design

The other example is a case in which a labeled CE design created by the L^{MA} design method is used in a questionnaire survey. Respondents were assumed to have been requested to select the most preferred from among three agricultural products and the option "none of these" (Figure 6). In this case, the region of origin attribute was 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. Therefore, the argument `attribute.names` in the function `Lma.design` is assigned by the attribute `Eco = c("Conv.", "More", "Most")` and the attribute `Price = c("1", "1.1", "1.2")`. The CE design was divided into 2 blocks. According to above mentioned condition, the function `Lma.design` was executed as follows (output is omitted):

```
R> 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)
```

The function `questionnaire` was executed with the argument as follows (output is partially omitted):

```
R> questionnaire(choice.experiment.design = des2)

Block 1

Question 1
      alt.1  alt.2   alt.3
Eco   "Most" "Conv." "More"
Price "1.2"  "1"     "1.1"

...

Question 9
      alt.1  alt.2  alt.3
Eco   "Most" "Most" "More"
Price "1"    "1.1"  "1"
```

```
Block 2

Question 1
      alt.1   alt.2  alt.3
Eco   "Conv." "More" "More"
Price "1.2"   "1.2"  "1"

...

Question 9
      alt.1  alt.2   alt.3
Eco   "Most" "Conv." "Conv."
Price "1.1"  "1.1"   "1.2"
```

Similar to the case in the first example, the responses to these CE questions were already created and stored in the package as the `syn.res2` data set as follows:

```
R> data("syn.res2")
R> syn.res2[1:3, ]

  ID BLOCK q1 q2 q3 q4 q5 q6 q7 q8 q9
1  1     1  4  1  3  4  4  4  2  3  1
2  2     2  1  2  4  2  1  3  3  1  1
3  3     1  1  2  4  3  1  2  1  4  3
```

Next, the design matrix, according to the CE design, is created and stored in the `desmat2` object as follows (output is omitted):

```
R> desmat2 <- make.design.matrix(choice.experiment.design = des2,
+    optout = TRUE, categorical.attributes = c("Eco"),
+    continuous.attributes = c("Price"), unlabeled = FALSE)
```

The function `make.dataset` combines these two data sets into one – `dataset2` – as follows (output is omitted):

```
R> dataset2 <- make.dataset(respondent.dataset = syn.res2,
+   choice.indicators =
+     c("q1", "q2", "q3", "q4", "q5", "q6", "q7", "q8", "q9"),
+   design.matrix = desmat2)
```

In the second example, the systematic component of utility for alternative i (= Reg_A, Reg_B, Reg_C) is modeled as follows (the systematic component of utility for the "none of these" option is normalized to zero):

$$V_{in} = ASC_i + b_{Morei}More_{in} + b_{Mosti}Most_{in} + b_{Pi}Price_{in}.$$

After executing the function `clogit` on the basis of the aforementioned model specification and the data set, the functions `gofm` and `mwtp` are executed as follows:

```
R> clogout2 <- clogit(RES ~ ASC1 + More1 + Most1 + Price1 + ASC2 + More2 +
+    Most2 + Price2 + ASC3 + More3 + Most3 + Price3 + strata(STR),
+    data = dataset2)
R> clogout2

Call:
clogit(RES ~ ASC1 + More1 + Most1 + Price1 + ASC2 + More2 + Most2 +
    Price2 + ASC3 + More3 + Most3 + Price3 + strata(STR), data = dataset2)


         coef exp(coef) se(coef)     z       p
ASC1    6.862  9.55e+02    1.073  6.40 1.6e-10
More1   0.324  1.38e+00    0.197  1.65 1.0e-01
Most1   0.558  1.75e+00    0.193  2.90 3.8e-03
Price1 -6.470  1.55e-03    0.988 -6.55 5.8e-11
ASC2    7.220  1.37e+03    1.133  6.37 1.9e-10
More2   0.590  1.80e+00    0.212  2.78 5.4e-03
Most2   0.877  2.40e+00    0.207  4.23 2.3e-05
Price2 -7.126  8.04e-04    1.049 -6.79 1.1e-11
ASC3    5.002  1.49e+02    1.117  4.48 7.5e-06
More3   0.471  1.60e+00    0.206  2.29 2.2e-02
Most3   0.484  1.62e+00    0.205  2.36 1.8e-02
Price3 -4.945  7.12e-03    1.025 -4.82 1.4e-06

Likelihood ratio test=167  on 12 df, p=0  n= 3600, number of events= 900

R> gofm(clogout2)

Rho-squared = 0.06674974
Adjusted rho-squared = 0.05713178
Number of coefficients = 12
Log likelihood at start = -1247.665
Log likelihood at convergence = -1164.384

R> mwtp(output = clogout2,
+    monetary.variables = c("Price1", "Price2", "Price3"),
+    nonmonetary.variables = list(
+      c("More1", "Most1"), c("More2", "Most2"), c("More3", "Most3")),
+    percentile.points = c(5, 95), seed = 987)

            MWTP         5%        95%
More1  0.0501243 -0.0000036  0.1048190
Most1  0.0862431  0.0368796  0.1442137
More2  0.0827328  0.0330651  0.1415383
Most2  0.1230702  0.0734755  0.1854955
More3  0.0952013  0.0254875  0.1878304
Most3  0.0979115  0.0297079  0.1882971
```

The coefficients of the variables – except for the coefficient of the variable `More1` – are significantly different from zero. Compared to the conventional cultivation method, the MWTPs for `More` and `Most` in each region – except for the MWTP for `More1` – are significantly positive.

5. Concluding remarks

The package **support.CEs** provides seven basic functions for supporting the implementation of CEs in R. We have also provided two hypothetical examples to demonstrate how these seven functions can be applied to consumers' valuations of an agricultural product. Although a certain amount of effort is necessary to conduct a survey and analysis in order to actualize appropriate applications for CEs, it is clear that the functions outlined in this paper will significantly contribute to the overall process, particularly for CE beginners. Since the package is devoted to supporting the basic implementation of CEs, additional efforts such as improving the functions in the package and developing new functions will undoubtedly be needed to conduct advanced CEs in R.

References

Aizaki H (2009). "Development of an Application Program for the Design and Analysis of Choice Experiments with R." *Kodo Keiryogaku*, **36**(1), 35–46. In Japanese with English summary.

Aizaki H (2012). ***support.CEs**: Basic Functions for Supporting an Implementation of Choice Experiments*. R package version 0.2-4, URL http://CRAN.R-project.org/package=support.CEs.

Aizaki H, Nishimura K (2008). "Design and Analysis of Choice Experiments Using R: A Brief Introduction." *Agricultural Information Research*, **17**(2), 86–94.

Bateman IJ, Carson RT, Day B, Hanemann M, Hanley N, Hett T, Jones-Lee M, Loomes G, Mourato S, Özdemiroğlu E, Pearce DW, Sugden R, Swanson J (2002). *Economic Valuation with Stated Preference Techniques: A Manual*. Edward Elger, Cheltenham, UK.

Ben-Akiva M, Lerman SR (1985). *Discrete Choice Analysis: Theory and Application to Travel Demand*. MIT Press, Cambridge, USA.

Bennett J, Blamey R (eds.) (2001). *The Choice Modelling Approach to Environmental Valuation*. Edward Elger, Cheltenham, UK.

Chrzan K, Orme B (2000). "An Overview and Comparison of Design Strategies for Choice-Based Conjoint Analysis." *Technical report*. URL http://www.sawtoothsoftware.com/download/techpap/desgncbc.pdf.

Dillman DA (2000). *Mail and Internet Surveys: The Tailored Design Method*. 2nd edition. John Wiley & Sons, New York, USA.

Greene WH (2011). *Econometric Analysis*. 7th edition. Prentice Hall, Upper Saddle River, USA.

Grömping U (2012). ***DoE.base**: Full Factorials, Orthogonal Arrays and Base Utilites for DoE Packages*. R package version 0.23-2, URL http://CRAN.R-project.org/package=DoE.base.

Hensher DA, Rose JM, Greene WH (2005). *Applied Choice Analysis: A Primer*. Cambridge University Press, Cambridge, UK.

Holmes TP, Adamowicz WL (2003). "Attribute-Based Methods." In PA Champ, KJ Boyle, TC Brown (eds.), *A Primer on Nonmarket Valuation*, pp. 171–219. Kluwer Academic Publishers, Dordrecht, NL.

Johnson FR, Kanninen B, Bingham M, Özdemir S (2007). "Experimental Design for Stated Choice Studies." In BJ Kanninen (ed.), *Valuing Environmental Amenities Using Stated Choice Studies: A Common Sense Approach to Theory and Practice*, pp. 159–202. Springer-Verlag, Dordrecht, NL.

Kanninen BJ (ed.) (2007). *Valuing Environmental Amenities Using Stated Choice Studies: A Common Sense Approach to Theory and Practice*. Springer-Verlag, Dordrecht, NL.

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

Louviere JJ, Hensher DA, Swait JD (2000). *Stated Choice Methods: Analysis and Applicaiton*. Cambridge University Press, Cambridge, UK.

Mangione TW (1995). *Mail Surveys: Improving the Quality*. Sage Publications, Thousand Oaks, USA.

R Development Core Team (2012). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL http://www.R-project.org/.

Ryan M, Gerard K, Amaya-Amaya M (eds.) (2008). *Using Discrete Choice Experiments to Value Health and Health Care*. Springer-Verlag, Dordrecht, NL.

Therneau T (2012). ***survival**: A Package for Survival Analysis in S*. R package version 2.36-12, URL http://CRAN.R-project.org/package=survival.

Wooldridge JM (2010). *Econometric Analysis of Cross Section and Panel Data*. 2nd edition. MIT Press, Cambridge, USA.

Affiliation:

Hideo Aizaki
Rural Development and Planning Division
Institute for Rural Engineering

National Agriculture and Food Research Organization
2-1-6 Kannondai, Tsukuba, Ibaraki 305-8609, Japan
E-mail: aizaki@affrc.go.jp

Journal of Statistical Software http://www.jstatsoft.org/
published by the American Statistical Association http://www.amstat.org/
Volume 50, Code Snippet 2 *Submitted:* 2010-05-04
September 2012 *Accepted:* 2012-08-22