Package ‘DCchoice’

September 30, 2020

Version 0.0.17

Date 2020-09-30

Title Analyzing Dichotomous Choice Contingent Valuation Data

Description Functions for analyzing dichotomous choice contingent valuation (CV) data. It provides functions for estimating parametric and nonparametric models for single-, one-and-one-half-, and double-bounded CV data. For details, see Aizaki et al. (2014) <doi:10.1201/b17292>.

Depends R (>= 3.5.0)

Imports stats, MASS, interval, Formula

Suggests Ecdat

URL http://www.agr.hokudai.ac.jp/spmur/

NeedsCompilation no

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

Author Tomoaki Nakatani [aut, cph] (original developer),
Hideo Aizaki [aut, cre] (code patches),
Kazuo Sato [ctb] (theoretical part of the manual)

Repository CRAN

Date/Publication 2020-09-30 08:10:02 UTC

R topics documented:

`DCchoice-package` *DCchoice: a package for analyzing dichotomous choice contingent valuation data*

Description

The package provides functions for analyzing single-, one-and-one-half-, and double-bounded dichotomous choice contingent valuation (CV) data

Details

This package provides functions for analyzing single-, one-and-one-half-, and double-bounded dichotomous choice contingent valuation (CV) data.

In the single-bounded dichotomous choice (SBDC) CV that was first proposed by Bishop and Heberlein (1979) respondents are requested to answer a question like the following:

If the environmental policy burdens you with USD X annually, would you agree or disagree to it?

This format, respondents only states "yes (I agree)" or "no (I disagree)," meaning whether their willingness to pay (WTP) is greater or lower than the bid (USD X) they are offered.

The double-bounded dichotomous choice (DBDC) CV was proposed by Hanemann (1985) and Carson (1985) to improve the efficiency of SBDC-CV. In the CV format, respondents are requested to answer the second (follow-up) question just after they answer the SBDC-CV style question (the first/initial question). An example of DBDC-CV questions is as follows ($Xl < X < Xh$):

First question
If the environmental policy burdens you with USD X annually, would you agree or disagree to it?

Second question for the respondents who agree to the policy in the first question
If the amount of tax is USD Xh, would you agree or disagree to it?

Second question for the respondents who disagree to the policy in the first question
If the amount of tax is USD Xl, would you agree or disagree to it?

In the DBDC-CV question, there are four possible response outcomes: (yes, yes); (yes, no); (no, yes); and (no, no). If the respondent i's answer is (yes, yes), the analyst can tell $WTPi > Xh$ ($WTPi$ is the WTP of the respondent i). Similarly, (yes, no) means $X < WTPi < Xh$, (no, yes) means $Xl < WTPi < X$, and (no, no) means $WTPi < Xl$.

One-and-one-half-bound dichotomous choice (OOHBDC) CV, which was developed by Cooper et al. (2002), is an intermediate CV format between SBDC-CV format and DBDC-CV format.

In the OOHBDC-CV survey, after answering the first SBDC-CV style question (the first stage), only respondents who satisfy certain conditions are requested to answer an additional SBDC-CV style question (the second stage). Details in the OOHBDC-CV survey are as follows:

Step 1) A list of bid ranges [BLj, BHj] (j = 1, 2, ..., J), where BLj < BHj, are decided: i.e., [BL1, BH1], [BL2, BH2], ..., and [BLJ, BHJ].

Step 2) One of the bid ranges is randomly presented to respondents (e.g., a bid range of [BL3, BH3] for j = 3).

Step 3) One of the two bids presented to the respondents is selected randomly (i.e., BL3 or BH3 in the case of step 2 example) and then the respondents are asked whether they would be willing to pay the amount of the bid selected (the first stage).

Step 4) The respondents are asked to answer the second stage if they satisfy either condition: (a) their answer in the first stage is "yes" when the lower bid is presented in the first stage, or (b) their answer in the first stage is "no" when the higher bid is presented in the first stage.

Therefore, there are six possible responses to the OOHBDC-CV survey: "no", "yes-no", and "yes-yes" when the lower bid is shown in the first stage; and "yes", "no-yes", and "no-no" when the higher bid is shown in the first stage. Refer to Cooper et al. (2002) for detailed explanation of OOHBDC-CV, including the example CV questions.

There are two ways of estimating WTP from the SBDC-, OOHBDC-, and DBDC-CV: parametric and nonparametric approaches. In this package, the functions `sbchoice`, `oohbchoice`, and `dbchoice`, which are based on the utility difference approach (Hanemann, 1984), are developed for the parametric approach to SBDC, OOHBDC, and DBDC data, respectively.

Confidence intervals for the estimates of WTPs are constructed by two methods. These are the Krinsky and Robb (1986, 1990)'s method and the bootstrap one. The former is implemented by `krCI` while the latter by `bootCI`.

Both of the methods rely on simulation techniques with different settings. Usually, a bootstrap method takes much longer time than the Krinsky and Robb's method does. It has been pointed out that each method has both advantages and disadvantages, see, for instance, the discussions in Hole (2007) and the references therein.

Functions for nonparametric approaches are also included in the package. `kristrom` (Kristrom, 1990) and `turnbull.sb` (Carson and Steinberg, 1990) are designed for analyses for SBDC data, whereas `turnbull.oohb` and `turnbull.db` (Carson and Hanemann, 2005) for OOHBDC and DBDC ones, respectively.

Note

DCchoice depends on **interval** (Fay and Shaw, 2010) package. It further depends on several packages, among which **Icens** (Gentleman and Vandal, 2011) package may not be installed by the GUI Package Installer accessible from the menu bar (Windows and Mac OS) because it is available only from Bioconductor.

To install **DCchoice** and other dependent packages simultaneously, use the `install.packages` function instead. See the **Examples** section for the code.

References

Aizaki H, Nakatani T, Sato K (2014). *Stated Preference Methods Using R.* CRC Press, Boca Raton FL. https://doi.org/10.1201/b17292.

Bishop RC, Heberlein TA (1979). "Measuring Values of Extra-Market Goods: Are Indirect Measures Biased?" *American Journal of Agricultural Economics*, **61**(5), 926–930.

Carson RT (1985). "Three Essays on Contingent Valuation". Dissertation, University of California Berkeley.

Carson RT, Hanemann WM (2005). "Contingent Valuation." in KG M\"aler, JR Vincent (eds.), *Handbook of Environmental Economics*. Elsevier, New York.

Carson RT, Steinberg D (1990). "Experimental Design for Discrete Choice Voter Preference Surveys." in *1989 Proceeding of the Survey Methodology Section of the American Statistical Association*, 821–822.

Cooper JC, Hanemann M, Signorello G (2002). "One-and-one-half-bound dichotomous choice contingent valuation", *The Review of Economics and Statistics*, **84**, 742–750.

Croissant Y (2011). **Ecdat***: Data Sets for Econometrics,* R package version 0.1-6.1, https://CRAN.R-project.org/package=Ecdat.

Fay MP, Shaw PA (2010). "Exact and Asymptotic Weighted Logrank Tests for Interval Censored Data: The **interval** R Package", *Journal of Statistical Software*, **36**(2), 1-34. https://www.jstatsoft.org/v36/i02/.

Gentleman R, Vandal A (2011). *Icens: NPMLE for Censored and Truncated Data*. R package version 1.24.0, https://CRAN.R-project.org/package=Icens.

Hanemann, WM (1984). "Welfare Evaluations in Contingent Valuation Experiments with Discrete Responses", *American Journal of Agricultural Economics*, **66**(2), 332–341.

Hanemann WM (1985). "Some Issues in Continuous- and Discrete-Response Contingent Valuation Studies." *Northeastern Journal of Agricultural Economics*, **14**, 5–13.

Hole AR (2007). "A Comparison of Approaches to Estimating Confidence Intervals for Willingness to Pay Measure." *Health Economics*, **16**, 827–840.

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

Krinsky I, Robb AL (1990). "On Approximating the Statistical Properties of Elasticities: A Correction." *The Review of Economics and Statistics*, **72**, 189–190.

Kristr\"om B (1990). "A Non-Parametric Approach to the Estimation of Welfare Measures in Discrete Response Valuation Studies." *Land Economics*, **66**(2), 135–139.

Examples

```
## Installation of DCchoice along with dependent packages is carried out
## by the following lines of code:
## Not run:
install.packages("DCchoice",
  repos = c("@CRAN@", "http://www.bioconductor.org/packages/release/bioc"),
  dep = TRUE)

## End(Not run)
## You may select a CRAN mirror a few times.
## For Mac and Unix/Linux users, add the option argument 'type="source"'.
```

AP *Albemarle-Pamlico sounds CVM data*

Description

Double-bounded dichotomous choice survey data for quality inprovements in the Albemarle-Pamlico sounds, North Carolina.

Usage

```
data(AP)
```

Format

a data frame containing 721 observations.

bid1 a vector of bids expressd in USD.

bid2 a vector of bids expressd in USD.

R1 a vector of binary dummies equal to 1 if the bid is accepted and 0 otherwise.

R2 a vector of binary dummies equal to 1 if the bid is accepted and 0 otherwise.

income a numeric vrector containing the annual household income in 1995 USD of the respondent

work a vector of binary dummies equal to 1 if the respondent is employed in fulltime and 0 otherwise.

age a numeric vector containing the age of the respondent.

female a vector of binary dummies equal to 1 if the respondent is female and 0 otherwise.

married a vector of binary dummies equal to 1 if the respondent is married and 0 otherwise.

Details

The original data are based on a telephone survey regarding quality improvements in the Albemarle-Pamlico sounds, North Carolina. The data were intensively analyzed, for instance, in Whitehead (1995) and Whitehead, et. al. (1998) in different contexts. Details of the survey can be found in the web site (see *Source* for the URL).

The original data have 1077 observations and include the bids and the responses of the double-bounded dichotomous choice survey. Various socio-demographic characteristics are also collected by the survey.

A subset of the data consisting of the responses to the CVM questions as well as minimum number of socio-demographic characteristics. In addition, observations with missing values were removed from the subset.

Source

The complete data and details can be obtained in the online version of Whitehead (2015) at `http://dx.doi.org/10.1016/j.dib.2015.01.006`.

The data are included here under kind permission from Professor John Whitehead, Appalachian State University.

References

Whitehead JC (1995). “Willingness to Pay for Quality Improvements: Comparative Statics and Interpretation of Contingent Valuation Results.” *Land Economics*, **71**(2), 207–215.

Whitehead JC (2015). “Albemarle-Pamlico Sounds Revealed and Stated Preference Data.” *Data in Brief*, **3**, 90–94. `http://dx.doi.org/10.1016/j.dib.2015.01.006`

Whitehead JC, Haab TC, Huang JC (1998). “Part-Whole Bias in Contingent Valuation: Will Scope Effects be Detected with Inexpensive Survey Methods?” *Southern Economic Journal*, **65**(1), 160–168.

---

bootCI — *Calculating confidence intervals for WTP using a nonparametric bootstrap method*

Description

This function calculates confidence intervals for WTP using the nonparametric bootstrap method.

Usage

```
bootCI(obj, nboot = 1000, CI = 0.95, individual = NULL)

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

Arguments

`obj`	an S3 class object `"dbchoice"` or `"sbchoice"`.
`nboot`	the number of bootstrap resampling.
`CI`	a percentile of the confidence intervals to be estimated.
`individual`	a data frame containing covariates that show an individual of which to estimate WTP.
`x`	an object of class `"bootCI"`.
`...`	optional arguments. Currently not in use.

Details

The bootstrap method resamples the data at our hands and repeatedly estimates the model with the bootstrapped data to formulate an empirical distribution of the associated WTP. This is a clear contrast with the method of Krinsky and Robb (1986, 1990) where the parameters are directly drawn from the multivariate normal distribution.

The upper and the lower bound of the interval is determined similarly to the case of the function `krCI`.

Hole (2007) conducted simulation experiments to compare the performance of the method of Krinsky and Robb (1986, 1990) with the bootstrap one.

A WTP of a specific individual (e.g., a representative respondent) can be estimated when assigning covariates to `individual`. See Example for details.

Value

The function `bootCI()` returns an object of S3 class `"bootCI"`. An object of `"bootCI"` is a list with the following components.

`out`	the output table with simulated confidence intervals as well as the four type of WTP estimates (mean, truncated mean, truncated mean with adjustment and median) from the ML estimation.
`mWTP`	a vector of simulated mean WTP. When |beta| < 1, this item is set to `-999`.
`tr.mWTP`	a vector of simulated mean WTP truncated at the maximum bid.
`adj.tr.mWTP`	a vector of simulated mean WTP truncated at the maximum bid with the adjustment.
`medWTP`	a vector of simulated median WTP.

When the parameter estimate on the bid does not satisfy the condition for the existence of the finite mean WTP (|beta|>1), the values of the lower and the upper bound of the confidence interval are coerced to set to `-999`.

The generic function `print()` is available for the object of class `"bootCI"` and displays the table of simulated confidence intervals.

The table contains the confidence intervals for the four types (mean, truncated mean, truncated mean with adjustment and median) of WTP from the ML estimation. The adjustment for the truncated mean WTP is implemented by the method of Boyle *et~al.*(1988).

Warning

It is time consuming (usually takes several minutes) to implement this function.

References

Boyle KJ, Welsh MP, Bishop RC (1988). "Validation of Empirical Measures of Welfare Change: Comment." *Land Economics*, **64**(1), 94–98.

Hole AR (2007). "A Comparison of Approaches to Estimating Confidence Intervals for Willingness to Pay Measure." *Health Economics*, **16**, 827–840.

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

Krinsky I, Robb AL (1990). "On Approximating the Statistical Properties of Elasticities: A Correction." *The Review of Economics and Statistics*, **72**, 189–190.

See Also

`krCI`, `dbchoice`, `sbchoice`

Examples

```
## See Examples in dbchoice and sbchoice.
```

Carson *Exxon Valdez Oil Spill CVM data*

Description

Contingency tables for the suggested bids and the number of respondents saying yes or no to the bids in the Exxon Valdez Oil Spill CVM survey.

Usage

```
data(CarsonSB)
data(CarsonDB)
```

Format

Both `CarsonSB` and `CarsonDB` are data frame objects of contingency tables.

For `CarsonSB`,

T1 a bid expressd in USD.

Y a number of respondents accepting the bid.

N a number of respondents not accepting the bid.

For `CarsonDB`,

T1 a first stage bid expressd in USD.

TU a second stage bid increased from the first one, expressd in USD.

TL a second stage bid decreased from the first one, expressd in USD.

yy a number of respondents accepting both the first and the second bids.

yn a number of respondents accepting only the first bid.

ny a number of respondents accepting only the second bid.

nn a number of respondents not accepting the first nor the second bids.

Details

Out of `CarsonSB` and `CarsonSB`, one may reconstruct the original yes/no type of data for 1043 observations. See the example for `CarsonSB`.

Source

`CarsonSB` and `CarsonDB` are reproduced from Tables A-15, A-16 and A-17 in Appendix C.1 of Carson et.al (1992).

The data are included under kind permission from Professor Richard T. Carson of University of California, San Diego.

References

Carson RT, Mitchell RC, Hanemann WM, Kopp RJ, Presser S, Ruud PA (1992). "A Contingent Valuation Study of Lost Passive Use Values Resulting from the Exxon Valdez Oil Spill." *Technical Report Report to the Attorney General of the State of Alaska*, Natural Resource Damage Assessment Inc. https://mpra.ub.uni-muenchen.de/6984/.

See Also

`ct2df`

Examples

```
## The following lines of code reconstruct the original yes/no type of data
## for 1043 observations. A data frame object sb.data consists of two variables,
## namely, bid1 and R1. The conversion into a simole data frame object can be
## done either manually or by using the \code{ct2df} function.
data(CarsonSB)

## Using the \code{ct2df} function
CarsonSB.dat <- ct2df(
  x    = CarsonSB,
  bid1 = "T1",
  y    = "Y",
  n    = "N",
  type = "single")
head(CarsonSB.dat)

# Manual conversion
```

```
n <- rowSums(CarsonSB[, -1])
sb.data <- data.frame(
  bid = c(rep(CarsonSB$T1[1], n[1]),
          rep(CarsonSB$T1[2], n[2]),
          rep(CarsonSB$T1[3], n[3]),
          rep(CarsonSB$T1[4], n[4])),
  R1 = c(rep(1, CarsonSB$Y[1]), rep(0, CarsonSB$N[1]),
         rep(1, CarsonSB$Y[2]), rep(0, CarsonSB$N[2]),
         rep(1, CarsonSB$Y[3]), rep(0, CarsonSB$N[3]),
         rep(1, CarsonSB$Y[4]), rep(0, CarsonSB$N[4]))
)
dim(sb.data)
head(sb.data)

## Double-bounded dichotomous choice CV format.
data(CarsonDB)
CarsonDB
CarsonDB.dat <- ct2df(
  x     = CarsonDB,
  bid1  = "T1",
  bid2h = "TU",
  bid2l = "TL",
  yy    = "yy",
  yn    = "yn",
  ny    = "ny",
  nn    = "nn",
  type  = "double")
head(CarsonDB.dat)

## An example of manual conversion is omitted.
## See Appendix 2.A of Aizaki, et. al. (2014).
```

`ct2df` *Convert a data frame in contingency-table format into a simple data frame of individual observations*

Description

A convinience function converting a data frame object in contingency-table format of bid(s) and responses of dichotomous choice CV into a simple data frame of individual observations. The outcome is suitable for the analysis using functions in the package.

Usage

```
ct2df(x, bid1 = "bid1", bid2h = "bidh", bid2l = "bidl",
    yy = "yy", yn = "yn", ny = "ny", nn = "nn", y = "y", n = "n", type = "double")
```

Arguments

x	a data frame object in contingency-table format containing bid(s) and responses
bid1	a character string showing the bid (for "single") or the bid in the first stage (for "double")
bid2h	a character string showing the second (higher) bid when respondents answer "Yes" in the first stage
bid2l	a character string showing the second (lower) bid when respondents answer "No" in the first stage
yy	a character string showing a number of respondents accepting both the first and the second bids
yn	a character string showing a number of respondents accepting only the first bid
ny	a character string showing a number of respondents accepting only the second bid
nn	a character string showing a number of respondents not accepting the first nor the second bids
y	a character string showing a number of respondents accepting the bid
n	a character string showing a number of respondents not accepting the bid
type	a character string setting the elicitation format, which takes one of "single" (single-bounded dichotomous choice format) or "double" (double-bounded dichotomous choice format)

Details

The function `ct2df` implements a conversion of a data frame containing bid(s) and responses regarding dichotomous choice CV in contingency-table format into a data frame suitable for use by the functions `sbchoice`, `dbchoice`, `kristrom`, `turnbull.sb`, and `turnbull.db`. See `CarsonSB` and `CarsonDB` for dataset in contingency-table format. Arguments from `bid2h` to `nn` are only valid for double-bounded dichotomous choice format, while arguments `y` and `n` are only valid for single-bounded dichotomous choice format.

See the examples, for usage in detail.

Value

The function returns a data frame, in which each row shows a single respondent. It contains the following variables.

For "single",

R1	a response to a bid: 1 for "Yes", 0 for "No"
bid1	the bid

For "double",

B1	a bid in the first stage
B2H	a (higher) bid in the second stage when the response is "Yes" in the first stage
B2L	a (lower) bid in the second stage when the response is "No" in the first stage

R	a combination of responses in the first and second stages, which takes yy for "Yes" and "Yes", yn for "Yes" and "No", ny for "No" and "Yes", or nn for "No" and "No"
R1	the response in the first stage, which takes 1 for "Yes", 0 for "No"
R2	the response in the second stage, which takes 1 for "Yes", 0 for "No"
bid1	the bid in the first stage
bid2	the bid in the second stage the respondent faced

References

Aizaki H, Nakatani T, Sato K (2014). Stated Preference Methods Using R. CRC Press, Boca Raton, FL.

See Also

sbchoice, dbchoice, kristrom, turnbull.sb, turnbull.db

Examples

```
# Single-bounded dichotomous choice CV format
data(CarsonSB)
CarsonSB
CarsonSB.dat <- ct2df(
  x    = CarsonSB,
  bid1 = "T1",
  y    = "Y",
  n    = "N",
  type = "single")
head(CarsonSB.dat)
summary(turnbull.sb(R1 ~ bid1, data = CarsonSB.dat))

# Double-bounded dichotomous choice CV format
data(CarsonDB)
CarsonDB
CarsonDB.dat <- ct2df(
  x     = CarsonDB,
  bid1  = "T1",
  bid2h = "TU",
  bid2l = "TL",
  yy    = "yy",
  yn    = "yn",
  ny    = "ny",
  nn    = "nn",
  type  = "double")
head(CarsonDB.dat)
summary(turnbull.db(R1 + R2 ~ bid1 + bid2, data = CarsonDB.dat))
```

dbchoice — *Parametric approach to analyze double-bounded dichotomous choice contingent valuation data*

Description

This function analyzes double-bounded dichotomous choice contingent valuation (CV) data on the basis of the utility difference approach.

Usage

```
dbchoice(formula, data, subset, na.action = na.omit, dist = "log-logistic",
         par = NULL, ...)

## S3 method for class 'dbchoice'
print(x, digits = max(3, getOption("digits") - 1), ...)

## S3 method for class 'dbchoice'
vcov(object, ...)

## S3 method for class 'dbchoice'
logLik(object, ...)
```

Arguments

`formula`	an object of S3 class "`formula`" and specifies the model structure.
`data`	a data frame containing the variables in the model formula.
`subset`	an optional vector specifying a subset of observations.
`na.action`	a function which indicates what should happen when the data contains `NA`s.
`dist`	a character string setting the error distribution in the model, which takes one of "`logistic`", "`normal`", "`log-logistic`", "`log-normal`" or "`weibull`".
`par`	a vector of initial parameters over which the optimization is carried out.
`x`	an object of class "`dbchoice`".
`digits`	a number of digits to display.
`object`	an object of class "`dbchoice`".
`...`	optional arguments. Currently not in use.

Details

The function `dbchoice()` implements an analysis of double-bounded dichotomous choice contingent valuation (CV) data on the basis of the utility difference approach (Hanemann, 1984). A generic call to `dbchoice()` is given by

`dbchoice(formula,data,dist = "log-logistic",...)`

The extractor function `summary()` is available for a `"dbchoice"` class object. See `summary.dbchoice` for details.

There are two functions available for computing the confidence intervals for the estimates of WTPs. `krCI` implements simulations to construct empirical distributions of the WTP while `bootCI` carries out nonparametric bootstrapping.

The argument `formula` defines the response variables and covariates. The argument `data` is mandatory where the data frame containing the variables in the model is specified. The argument `dist` sets the error distribution. Currently, one of `"logistic"`, `"normal"`, `"log-logistic"`, `"log-normal"`, or `"weibull"` is available. The default value is `dist = "log-logistic"`, so that it may be omitted if the user wants to estimate a model with log-logistic error distribution.

The difference between normal and log-normal models or between logistic or log-logistic ones is how the bid variable is incorporated into the model to be estimated. For the Weibull model, the bid variable must be entered in the natural log. Therefore, the user must be careful in defining the model formula that is explained in details below.

A typical structure of the formula for `dbchoice()` is defined as follows:

`R1 + R2 ~ (the names of the covariates) | BD1 + BD2`

The formula is an object of class `"formula"` and specifies the model structure. It has to be written in a symbolic expression in R. The formula consists of three parts. The first part, the left-hand side of the tilde sign (~), must contain the response variables for the suggested prices in the first and the second stage of CV questions. In the example below, `R1` denotes a binary or two-level factor response variable for a bid in the first stage and `R2` for a bid in the second stage. Each of `R1` and `R2` contains `"Yes"` or `"No"` to the bid or `1` for `"Yes"` and `0` for `"No"`.

The covariates are defined in the second part in the place of `(the names of the covariates)`. Each covariate is connected with the arithmetic operator `+` and `(the names of the covariates)` in the above syntax should be replaced with `var1 + var2` and the like. The plus sign is nothing to do with addition of the two variables in the symbolic expression. When the covariate contains only a constant term, a value of `1` is set as the covariate (that is, `R1 + R2 ~ 1 | BD1 + BD2`)

The last part starts after the vertical bar (`|`). The names of the two variables (`BD1` and `BD2`) containing suggested prices in the first and second stage of double-bounded dichotomous choice CV question are specified in this part. The two variables are also connected with the arithmetic operator (`+`).

According to the structure of the formula, a data set (data frame) consists of three parts. An example of the data set is as follows (`sex`, `age`, and `income` are respondents characteristics and assumed to be covariates):

R1	R2	sex	age	income	BD1	BD2
Yes	Yes	Male	20	Low	100	250
Yes	No	Male	30	Low	500	1000
...						

The second bid in the double-bounded dichotomous choice CV question is larger or lower than the first bit according to the response to the first stage: if the response to the first stage is `"Yes"`, the second bid is larger than the first bid; if the response is `"No"`, the second bid is lower than the first bid. In the example above, `BD2` is set as the second bid according to each respondent faced in the second stage. However, the followings style of data set is frequently prepared:

```
R1    R2    sex   age   income   BD1   BD2H   BD2L
```

Yes	Yes	Male	20	Low	100	250	50
Yes	No	Male	30	Low	500	1000	250
...							

BD2H is the second (higher) bid when the respondent answers "Yes" in the first stage; BD2L is the second (lower) bid when the respondent answers "No" in the first stage. In this case, the users have to convert BD2H and BD2L into BD2 (see the section "Examples").

The function dbchoice() analyzes double-bounded dichotomous choice CV data using the function optim on the basis of the initial coefficients that are estimated from a binary logit model analysis of the first-stage CV responses (the binary logit model is estimated internally by the function glm with the argument family = binomial(link = "logit")).

Nonparametric analysis of double-bounded dichotomous choice data can be done by turnbull.db. A single-bounded analogue of dbchoice is called sbchoice.

Value

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

f.stage	a list of components returned from the function glm based on the responses to the first CV question. The coefficient estimates of the first stage estimation is used as the initial coefficients for full analysis using the function optim. If par is not NULL, the supplied vector is returned.
dbchoice	a list of components returned from the function optim.
coefficients	a named vector of estimated coefficients.
call	the matched call.
formula	the formula supplied.
Hessian	an estimate of the Hessian. See also Hessian in optim.
distribution	a character string showing the error distribution used.
loglik	a value of the log likelihood at the estimates.
convergence	an logical code: TRUE means a successful convergence.
niter	a vector of two integers describing the number of calls to the object function and the numerical gradient, respectively. See also counts in optim.
nobs	a number of observations.
covariates	a named matrix of the covariates used in the model.
bid	a named matrix of the bids used in the model.
yn	a named matrix of the responses to the initial and follow-up CV questions used in the model.
data.name	the data matrix.
terms	terms
contrast	contrasts used for factors
xlevels	levels used for factors

References

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

Carson RT, Hanemann WM (2005). “Contingent Valuation.” in KG M\"aler, JR Vincent (eds.), *Handbook of Environmental Economics*. Elsevier, New York.

Croissant Y (2011). **Ecdat***: Data Sets for Econometrics,* R package version 0.1-6.1, https://CRAN.R-project.org/package=Ecdat.

Hanemann, WM (1984). “Welfare Evaluations in Contingent Valuation Experiments with Discrete Responses”, *American Journal of Agricultural Economics*, **66**(2), 332–341.

Hanemann M, Kanninen B (1999). “The Statistical Analysis of Discrete-Response CV Data.”, in IJ Bateman, KG Willis (eds.), *Valuing Environmental Preferences: Theory and Practice of the Contingent Valuation Methods in the US, EU, and Developing Countries*, 302–441. Oxford University Press, New York.

Hanemann WM, Loomis JB, Kanninen BJ (1991). “Statistical Efficiency of Double-Bounded Dichotomous Choice Contingent Valuation.” *American Journal of Agricultural Economics*, **73**(4), 1255–1263.

See Also

summary.dbchoice, krCI, bootCI, sbchoice, turnbull.db, NaturalPark, glm, optim, formula

Examples

```
## Examples are based on a data set NaturalPark in the package
## Ecdat (Croissant 2011): DBDCCV style question for measuring
## willingness to pay for the preservation of the Alentejo Natural
## Park. The data set (dataframe) contains seven variables:
## bid1 (bid in the initial question), bidh (higher bid in the follow-up
## question), bidl (lower bid in the follow-up question), answers
## (response outcomes in a factor format with four levels of "nn",
## "ny", "yn", "yy"), respondents' characteristic variables such
## as age, sex and income (see NaturalPark for details).
data(NaturalPark, package = "Ecdat")
head(NaturalPark)

## The variable answers are converted into a format that is suitable for the
## function dbchoice() as follows:
NaturalPark$R1 <- ifelse(substr(NaturalPark$answers, 1, 1) == "y", 1, 0)
NaturalPark$R2 <- ifelse(substr(NaturalPark$answers, 2, 2) == "y", 1, 0)

## We assume that the error distribution in the model is a
## log-logistic; therefore, the bid variables bid1 is converted
## into LBD1 as follows:
NaturalPark$LBD1 <- log(NaturalPark$bid1)

## Further, the variables bidh and bidl are integrated into one
## variable (bid2) and the variable is converted into LBD2 as follows:
NaturalPark$bid2 <- ifelse(NaturalPark$R1 == 1, NaturalPark$bidh, NaturalPark$bidl)
```

```
NaturalPark$LBD2 <- log(NaturalPark$bid2)

## The utility difference function is assumed to contain covariates (sex, age, and
## income) as well as two bid variables (LBD1 and LBD2) as follows:
fmdb <- R1 + R2 ~ sex + age + income | LBD1 + LBD2

## Not run:
## The formula may be alternatively defined as
fmdb <- R1 + R2 ~ sex + age + income | log(bid1) + log(bid2)

## End(Not run)

## The function dbchoice() with the function fmdb and the dataframe
## NP is executed as follows:
NPdb <- dbchoice(fmdb, data = NaturalPark)
NPdb
NPdbs <- summary(NPdb)
NPdbs

## The confidence intervals for these WTPs are calculated using the
## function krCI() or bootCI() as follows:
## Not run:
krCI(NPdb)
bootCI(NPdb)

## End(Not run)
## The WTP of a female with age = 5 and income = 3 is calculated
## using function krCI() or bootCI() as follows:
## Not run:
krCI(NPdb, individual = data.frame(sex = "female", age = 5, income = 3))
bootCI(NPdb, individual = data.frame(sex = "female", age = 5, income = 3))

## End(Not run)

## The variable age and income are deleted from the fitted model,
## and the updated model is fitted as follows:
update(NPdb, .~. - age - income |.)

## The bid design used in this example is created as follows:
bid.design <- unique(NaturalPark[, c(1:3)])
bid.design <- log(bid.design)
colnames(bid.design) <- c("LBD1", "LBDH", "LBDL")
bid.design
## Respondents' utility and probability of choosing Yes-Yes, Yes-No,
## No-Yes, and No-No under the fitted model and original data are
## predicted as follows:
head(predict(NPdb, type = "utility", bid = bid.design))
head(predict(NPdb, type = "probability", bid = bid.design))
## Utility and probability of choosing Yes for a female with age = 5
## and income = 3 under bid = 10 are predicted as follows:
predict(NPdb, type = "utility",
    newdata = data.frame(sex = "female", age = 5, income = 3, LBD1 = log(10)))
predict(NPdb, type = "probability",
```

```
    newdata = data.frame(sex = "female", age = 5, income = 3, LBD1 = log(10)))

## Plot of probabilities of choosing yes is drawn as drawn as follows:
plot(NPdb)
## The range of bid can be limited (e.g., [log(10), log(20)]):
plot(NPdb, bid = c(log(10), log(20)))
```

KR *Kristr\"om's single-bounded dichotomous choice CVM data*

Description

A single-bounded dichotomous choice CVM data analyzed in Kristr\"om (1990).

Usage

```
data(KR)
```

Format

A data frame of single-bounded dichotomous choice contingent valuation survey data. The number of observations is 562.

bid1 a vector of bids expressd in SEK.

R1 a vector of binary dummies equal to 1 if the bid is accepted and 0 otherwise.

Details

The data consist of the responses to the single-bounded dichotomous choice survey for a sample of 562 Swedes regarding preservation of the eleven virgin forests in Sweden. See Kristr\"om (1990) for more details.

Source

The data are used in Kristr\"om (1990).

The data are bundled in this package under kind permission from Professor Bengt Kristr\"om, Swedish University of Agricultural Sciences.

References

Kristr\"om B (1990). "A Non-Parametric Approach to the Estimation of Welfare Measures in Discrete Response Valuation Studies." *Land Economics*, **66**(2), 135–139.

`krCI` *Calculating confidence intervals for WTP using a parametric simulation*

Description

This function calculates confidence intervals for WTP using the method of Krinsky and Robb (1986, 1990).

Usage

```
krCI(obj, nsim = 1000, CI = 0.95, individual = NULL)

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

Arguments

`obj`	an S3 class object "`dbchoice`" or "`sbchoice`".
`nsim`	the number of draws of the parameters.
`CI`	a percentile of the confidence intervals to be estimated.
`individual`	a data frame containing covariates that show an individual of which to estimate WTP.
`x`	an object of class "`krCI`".
`...`	optional arguments. Currently not in use.

Details

In the method of Krinsky and Robb (1986, 1990), a set of parameters is drawn `nsim` times from a multivariate normal distribution with a vector of the parameter estimates as a mean and the variance-covariance matrix of the parameter estimates. Then, various WTPs are computed for each draw of simulated parameters. As a result, we are able to build an empirical distribution of the WTPs concerned, and hence the confidence intervals. For each WTP, and when `nsim = 1000`, the lower and the upper bound of the 95% confidence interval (`CI = 0.95`) correspond to the 26th and the 975th sorted estimates, respectively.

Confidence intervals based on the bootstrap method are calculated by `bootCI`.

Hole (2007) conducted simulation experiments to compare the performance of the method of Krinsky and Robb (1986, 1990) with the bootstrap one.

A WTP of a specific individual (e.g., a representative respondent) can be estimated when assigning covariates to `individual`. See Example for details.

Value

The function `krCI()` returns an object of S3 class `"krCI"`. An object of `"krCI"` is a list with the following components.

- `out` — the output table with simulated confidence intervals as well as the four type of WTP estimates (mean, truncated mean, truncated mean with adjustment and median) from the ML estimation.
- `mWTP` — a vector of simulated mean WTP. When |beta| < 1, this item is set to `-999`.
- `tr.mWTP` — a vector of simulated mean WTP truncated at the maximum bid.
- `adj.tr.mWTP` — a vector of simulated mean WTP truncated at the maximum bid with the adjustment.
- `medWTP` — a vector of simulated median WTP.

When the parameter estimate on the bid does not satisfy the condition for the existence of the finite mean WTP (|beta|>1), the values of the lower and the upper bound of the confidence interval are coerced to set to `-999`.

The table contains the confidence intervals for the four types (mean, truncated mean, truncated mean with adjustment and median) of WTP estimate from the ML estimation. The adjustment for the truncated mean WTP is implemented by the method of Boyle *et~al.*(1988).

The generic function `print()` is available for the object of class `"krCI"` and displays the table of simulated confidence intervals.

References

Boyle KJ, Welsh MP, Bishop RC (1988). "Validation of Empirical Measures of Welfare Change: Comment." *Land Economics*, **64**(1), 94–98.

Hole AR (2007). "A Comparison of Approaches to Estimating Confidence Intervals for Willingness to Pay Measure." *Health Economics*, **16**, 827–840.

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

Krinsky I, Robb AL (1990). "On Approximating the Statistical Properties of Elasticities: A Correction." *The Review of Economics and Statistics*, **72**, 189–190.

See Also

`bootCI`, `dbchoice`, `sbchoice`

Examples

```
## See Examples in dbchoice and sbchoice.
```

`kristrom` *The Kristr\"om's nonparametric approach to analyze single-bounded dichotomous choice contingent valuation data*

Description

This function analyzes single-bounded dichotomous choice contingent valuation (CV) data on the basis of the Kristr\"om's nonparametric method.

Usage

```
kristrom(formula, data, subset)

## S3 method for class 'kristrom'
print(x, digits = 4, ...)
```

Arguments

`formula`	a formula specifying the model structure.
`data`	a data frame containing the variables in the model formula.
`subset`	an optional vector specifying a subset of observations.
`x`	an object of class `"kristrom"`.
`digits`	a number of digits to display.
`...`	optional arguments. Currently not in use.

Details

The function `kristrom()` analyzes single-bounded dichotomous choice contingent valuation (CV) data on the basis of Kristr\"om's nonparametric method (Kristr\"om 1990).

The argument `formula` defines the response variables and bid variables. The argument `data` is set as a data frame containing the variables in the model.

A typical structure of the formula for `kristrom()` is defined as follows:

`R1 ~ BD1`

The formula consists of two parts. The first part, the left-hand side of the tilde sign (~), must contain the response variable (e.g., `R1`) for the suggested prices in the CV questions. The response variable contains `"Yes"` or `"No"` to the bid or `1` for `"Yes"` and `0` for `"No"`. The other part, which starts after the tilde sign, must contain a bid variable (e.g., `BD1`) containing suggested prices in the CV question.

The structure of data set which assigned to the argument data is the same as that in case of `dbchoice()`. See `dbchoice` for details in the data set structure.

Value

The function kristrom() returns an object of S3 class "kristrom". An object of "kristrom" is a list with the following components.

tab.dat	a matrix describing the number of respondents who answered "yes" to CV question, total number of respondents, and ratio of respondents who answered "yes" among the total number of respondents for each value of the suggested bids.
M	the number of rows of tab.dat.
adj.p	a vector describing the probability of a yes-answer to the suggested bid, which is the same as the last column of tab.dat.
nobs	the number of observations.
unq.bid	a vector of the unique bids.
estimates	a matrix of the estimated Kristr\"om's survival probabilities.

The generic function print() is available for fitted model object of class "kristrom" and displays the estimated Kristr\"om's survival probabilities.

The extractor function summary() is used to display the estimated Kristr\"om's survival probabilities as well as three types of WTP estimates (Kaplan-Meier and Spearman-Karber mean, and median estimates). Note that the Spearman-Karber mean estimate is computed upto the X-intercept.

A graph of the estimated empirical survival function is depicted by plot(). See plot.kristrom for details.

turnbull.sb is an alternative nonparametric method for analyzing single-bounded dichotomous choice data. A parametric analysis can be done by sbchoice.

References

Croissant Y (2011). **Ecdat**: *Data Sets for Econometrics,* R package version 0.1-6.1, https://CRAN.R-project.org/package=Ecdat.

Kristr\"om B (1990). "A Non-Parametric Approach to the Estimation of Welfare Measures in Discrete Response Valuation Studies." *Land Economics*, **66**(2), 135–139.

See Also

plot.kristrom, NaturalPark, turnbull.sb, sbchoice

Examples

```
## Examples for kristrom() are also based on a data set NaturalPark in the package
## Ecdat (Croissant 2011): so see the section Examples in the dbchoice() for details.
data(NaturalPark, package = "Ecdat")

## The variable answers are converted into a format that is suitable for the function
## kristrom() as follows:
NaturalPark$R1 <- ifelse(substr(NaturalPark$answers, 1, 1) == "y", 1, 0)

## The formula is defined as follows:
fmks <- R1 ~ bid1
```

```
## The function kristrom() with the function fmks and the data frame NP
## is executed as follows:
NPks <- kristrom(fmks, data = NaturalPark)
NPks
NPkss <- summary(NPks)
NPkss
plot(NPks)
```

`oohbchoice` *Parametric approach to analyze one-and-one-half-bound dichotomous choice contingent valuation data*

Description

This function analyzes one-and-one-half-bound dichotomous choice contingent valuation (CV) data on the basis of the utility difference approach.

Usage

```
oohbchoice(formula, data, subset, na.action = na.omit, dist = "log-logistic",
           par = NULL, ...)
```

Arguments

`formula`	an object of S3 class `"Formula"` and specifies the model structure.
`data`	a data frame containing the variables in the model formula.
`subset`	an optional vector specifying a subset of observations.
`na.action`	a function which indicates what should happen when the data contains `NA`s.
`dist`	a character string setting the error distribution in the model, which takes one of `"logistic"`, `"normal"`, `"log-logistic"`, `"log-normal"` or `"weibull"`.
`par`	a vector of initial parameters over which the optimization is carried out.
`...`	optional arguments. Currently not in use.

Details

One-and-one-half-bound dichotomous choice contingent valuation (OOHBDC-CV), which was developed by Cooper et al. (2002), is an intermediate CV format between single-bounded dichotomous choice (SBDC) CV format and double-bounded dichotomous choice (DBDC) CV format.

On the basis of an example of environmental valuation study, we will explain differences in question format among SBDC-CV, DBDC-CV, and OOHBDC-CV below.

In any of three CV surveys, two situations are firstly explained to the respondents: the current situation and the improved situation where an environmental improvement plan is assumed to be implemented. Question following the explanation of situations differs according to CV types.

In an SBDC-CV survey, after the explanation of situation mentioned above, the respondents are asked whether they would be willing to pay a certain amount of money toward the implementation of the plan. Therefore, there are two possible responses to the SBDC-CV survey: "yes (agree)," and "no (disagree)." The amounts (bids) that respondents are requested to contribute toward the plan are listed in advance. Each respondent is requested to answer a question randomly assigned with one of the listed bids.

In a DBDC-CV survey, the CV question consists of two stages: after answering the SBDC-CV style question mentioned above (the first stage), the respondents are also asked to answer an additional SBDC-CV style question (the second stage). The bid in the second stage varies according the response in the first stage: a higher bid is displayed in the second stage if the response in the first stage is "yes," whereas a lower bid is displayed when the response in the first stage is "no." Therefore, there are four possible responses to the DBDC-CV survey: "yes-yes" ("yes" in the both stages), "yes-no" ("yes" and "no" in the first and second stages, respectively), "no-yes" ("no" and "yes" in the first and second stages, respectively), and "no-no" ("no" in the both stages).

In the OOHBDC-CV survey, after answering the first SBDC-CV style question (the first stage), only respondents who satisfy certain conditions are requested to answer an additional SBDC-CV style question (the second stage). Details in the OOHBDC-CV survey are as follows: Step 1) A list of bid ranges [BLj, BHj] (j = 1, 2, ..., J), where BLj < BHj, are decided: i.e., [BL1, BH1], [BL2, BH2], ..., and [BLJ, BHJ]. Step 2) One of the bid ranges is randomly presented to respondents (e.g., a bid range of [BL3, BH3] for j = 3). Step 3) One of the two bids presented to the respondents is selected randomly (i.e., BL3 or BH3 in the case of step 2 example) and then the respondents are asked whether they would be willing to pay the amount of the bid selected (the first stage). Step 4) The respondents are asked to answer the second stage if they satisfy either condition: (a) their answer in the first stage is "yes" when the lower bid is presented in the first stage, or (b) their answer in the first stage is "no" when the higher bid is presented in the first stage. Therefore, there are six possible responses to the OOHBDC-CV survey: "no", "yes-no", and "yes-yes" when the lower bid is shown in the first stage; and "yes", "no-yes", and "no-no" when the higher bid is shown in the first stage. Refer to Cooper et al. (2002) for detailed explanation of OOHBDC-CV, including the example CV questions.

The function `oohbchoice()` implements an analysis of OOHBDC-CV responses (data) on the basis of the utility difference approach (Hanemann, 1984).

The function returns an object of S3 class `oohbchoice` (see below for details), which inherits from an S3 class `dbchoice`. The generic functions for the S3 class `dbchoice` such as `print()`, `summary()`, `vcov()`, `logLik()`, `plot()`, and `update()`, are available for the S3 class `oohbchoice`. In addition, the two functions `krCI()` and `bootCI()` are available to compute the confidence intervals for the estimates of willingness-to-pays (WTPs): `krCI()` implements simulations to construct empirical distributions of the WTP, while `bootCI()` carries out nonparametric bootstrapping (see the package **DCchoice** for details).

Although `oohbchoice()` has six arguments, a basic generic call to `oohbchoice()` is given as follows:

`oohbchoice(formula,data,dist = "log-logistic")`

The argument `formula` defines the response variables and covariates (see below for details on the formula). The argument `data` specifies the data frame containing the variables in the model. The argument `dist` sets the error distribution: one of `"logistic"`, `"normal"`, `"log-logistic"` (default value), `"log-normal"`, or `"weibull"` is available. The difference between normal and log-normal models or between logistic or log-logistic ones is how the bid variable is incorporated into the

model to be estimated. For the Weibull model, the bid variable must be entered in the natural log. Therefore, the user must be careful in defining the model formula that is explained in details below.

A typical structure of the formula for `oohbchoice()` is defined as follows:

`R1 + R2 ~ (the names of the covariates) | BL + BH`

The formula is an object of S3 class `Formula` and specifies the model structure. It has to be written in a symbolic expression in R. The formula consists of three parts as follows.

The first part, the left-hand side of the tilde sign (~), must contain the response variables for the suggested prices in the first and the second stage of CV questions. In the example below, `R1` denotes a binary or two-level factor response variable for a bid in the first stage and `R2` for a bid in the second stage. `R1` contains `yes` or `no` to the bid in the first stage or `1` for `yes` and `0` for `no`. `R2` contains `yes`, `no`, `none` to the bid in the second stage or `1` for `yes`, `0` for `no`, and `-9` for `none`. The value of `none` (`-9`) means that the respondents have no second stage: the respondents are asked to answer the second stage question only if they satisfy either condition: (a) they answer `yes` in the first stage when the lower bid is shown in the first stage, or (b) they answer `no` in the first stage when the higher bid is shown in the first stage.

The covariates are defined in the second part of the formula in the place of (the names of the covariates). Each covariate is connected with the arithmetic operator `+` and (the names of the covariates) in the above syntax should be replaced with `var1 + var2` and the like. The plus sign is nothing to do with addition of the two variables in the symbolic expression. When the covariate contains only a constant term, a value of `1` is set as the covariate: `R1 + R2 ~ 1 | BL + BH` (see the examples section below)

The last part of the formula starts after the vertical bar (`|`). The names of the two variables (`BL` and `BH`) containing suggested lower and higher prices in OOHBDC-CV question are specified in this part. The two variables are also connected with the arithmetic operator (`+`).

According to the structure of the formula, a data set (data frame) consists of three parts. An example of the data set (first six rows) is as follows (`gender` and `age` are respondents' characteristics and assumed to be covariates):

```
id   R1    R2  gender  age    BL    BH
 1   no  none    male   51  2000  2500
 2  yes    no    male   30   500  1000
 3  yes   yes  female   25   500  1000
 4  yes  none    male   48  1000  1500
 5   no   yes    male   60  1000  1500
 6   no    no  female   34  2500  3000
```

Respondent 1 faced a bid range [`2000`, `2500`]; respondents 2 and 3 faced a bid range [`500`, `1000`]; respondents 4 and 5 faced a bid range [`1000`, `1500`]; and respondent 6 faced [`2500`, `3000`]. Respondent 1 answered `no` in the first stage of CV question and had no the second stage; respondent 2 answered `yes` and `no` in the first and second stage, respectively; respondent 3 answered `yes` and `yes` in the both stages; respondent 4 answered `yes` in the first stage and had no the second stage; respondent 5 answered `no` and `yes` in the first and second stage; and respondent 6 answered `no` in the both stages.

Note that `BL` and `BH` are NOT the first stage bid and the second stage bid, respectively. The function `oohbchoice()` understands which bids (`BL` and `BH`) are presented in the first stage and second stage, respectively, on the basis of values of variables `R1` and `R2`.

Nonparametric analysis of OOHBDC-CV data can be done by `turnbull.oohb`.

Value

The function returns an S3 class object `oohbchoice`, which inherits from the S3 class `dbchoice`. See `dbchoice()` for the details on the S3 object `dbchoice`.

Acknowledgments

We would like to thank Dr. Joseph C. Cooper and Dr. Giovanni Signorello for their kindness.

References

Cooper JC, Hanemann M, Signorello G (2002). "One-and-one-half-bound dichotomous choice contingent valuation", *The Review of Economics and Statistics*, **84**, 742–750.

Hanemann WM (1984). "Welfare Evaluations in Contingent Valuation Experiments with Discrete Responses", *American Journal of Agricultural Economics*, **66**(2), 332–341.

See Also

`summary.dbchoice`, `oohbsyn`, `krCI`, `bootCI`, `turnbull.oohb`, `Formula`

Examples

```
## See oohbsyn.
```

`oohbsyn` *Synthetic data set for oohbchoice()*

Description

Dataset created artificially for the examples section of the function `oohbhoice()`.

Usage

```
data(oohbsyn)
```

Format

A data frame with 80 observations on the following variables.

id a vector of the identification number of the respondent.

gender a vector containing the gender of the respondent, taking `male` or `female`.

age a vector containing the age of the respondent.

BL a vector of lower bid.

BH a vector of higher bid.

R1 a vector of response to the first stage CV question, taking a value of 1 if the bid is accepted, and 0 otherwise.

R2 a vector of response to the second stage CV question, taking a value of 1 if the bid is accepted, 0 if the bid is not accepted, and -9 if the respondent has no the second stage question.

See Also

oohbchoice, turnbull.oohb

Examples

```
## Parametric model
data(oohbsyn)
oohb1 <- oohbchoice(R1 + R2 ~ 1 | log(BL) + log(BH), data = oohbsyn)
summary(oohb1)
oohb2 <- oohbchoice(R1 + R2 ~ age + gender | log(BL) + log(BH), data = oohbsyn)
summary(oohb2)

## Non-parametric model
oohb3 <- turnbull.oohb(R1 + R2 ~ BL + BH, data = oohbsyn)
summary(oohb3)
plot(oohb3)
```

---

plot.dbchoice *Plotting dbchoice objects*

Description

Plotting method for objects of class "dbchoice".

Usage

```
## S3 method for class 'dbchoice'
plot(x, type = NULL, main = NULL, sub = NULL, xlab = NULL,
ylab = NULL, lwd = NULL, lty = NULL, xlim = NULL, ylim = NULL, bid = NULL, ...)
```

Arguments

x	an object of class "kristrom".
type	type of plot.
main	the main title of the plot. If unspecified, no main title is displayed.
sub	the sub-title of the plot. If unspecified, no sub-title is displayed.
xlab	the x label of the plot. If missing, xlab = "Bid" is used. Setting xlab = "" displays no x label.
ylab	the y label of the plot. If missing, ylab = "Probability of selecting yes" is used. Setting ylab = "" displays no y label.

`lwd`	the line width for the plot. If missing, `lwd = 3` is used.
`lty`	the line type for the plot. If missing, `lty = 1` is used.
`xlim`	the x limits of the plot.
`ylim`	the y limits of the plot.
`bid`	the bid limits should be drawn. If missing, the minimum and maximum values of the bid variable(s) in the original dataset is used.
`...`	optional arguments.

Details

The function `plot()` draws choice probabilities of yes according to the range of bid (covariates are set on average). Choice probabilities are calculated according to the relevant single-bounded dichotomous choice model.

See Also

`dbchoice`

Examples

```
## See Examples in dbchoice.
```

`plot.kristrom` *Plotting* `kristrom` *objects*

Description

Plotting method for objects of class `"kristrom"`. The empirical survival curve is plotted.

Usage

```
## S3 method for class 'kristrom'
plot(x, main = NULL, sub = NULL, xlab = NULL, ylab = NULL,
    lwd = NULL, lty = NULL, ...)
```

Arguments

`x`	an object of class `"kristrom"`.
`main`	the main title of the plot. If unspecified, no main title is displayed.
`sub`	the sub-title of the plot. If unspecified, no sub-title is displayed.
`xlab`	the x label of the plot. If missing, `xlab = "Bid"` is used. Setting `xlab = ""` displays no x label.
`ylab`	the y label of the plot. If missing, `ylab = "Survival Probability"` is used. Setting `ylab = ""` displays no y label.
`lwd`	the line width for the plot. If missing, `lwd = 3` is used.
`lty`	the line type for the plot. If missing, `lty = 1` is used.
`...`	optional arguments. Currently not in use.

See Also

kristrom, summary.kristrom

---

plot.sbchoice *Plotting sbchoice objects*

Description

Plotting method for objects of class "sbchoice".

Usage

```
## S3 method for class 'sbchoice'
plot(x, type = NULL, main = NULL, sub = NULL, xlab = NULL,
ylab = NULL, lwd = NULL, lty = NULL, xlim = NULL, ylim = NULL, bid = NULL, ...)
```

Arguments

x	an object of class "kristrom".
type	type of plot.
main	the main title of the plot. If unspecified, no main title is displayed.
sub	the sub-title of the plot. If unspecified, no sub-title is displayed.
xlab	the x label of the plot. If missing, xlab = "Bid" is used. Setting xlab = "" displays no x label.
ylab	the y label of the plot. If missing, ylab = "Probability of selecting yes" is used. Setting ylab = "" displays no y label.
lwd	the line width for the plot. If missing, lwd = 3 is used.
lty	the line type for the plot. If missing, lty = 1 is used.
xlim	the x limits of the plot.
ylim	the y limits of the plot.
bid	the bid limits should be drawn. If missing, the minimum and maximum values of the bid variable(s) in the original dataset is used.
...	optional arguments.

Details

The function plot() draws choice probabilities of yes according to the range of bid (covariates are set on average).

See Also

sbchoice

Examples

```
## See Examples in sbchoice.
```

`plot.turnbull` *Plotting* `turnbull` *objects*

Description

Plotting method for objects of class "turnbull". The empirical survival curve and confidence interval (if computed) are plotted.

Usage

```
## S3 method for class 'turnbull'
plot(x, main = NULL, sub = NULL, xlab = NULL, ylab = NULL,
        lwd = NULL, lty = NULL, plotCI = FALSE, ltyCI = 5, ...)
```

Arguments

x	an object of class "turnbull".
main	the main title of the plot. If unspecified, no main title is displayed.
sub	the sub-title of the plot. If unspecified, no sub-title is displayed.
xlab	the x label of the plot. If missing, xlab = "Bid" is used. Setting xlab = "" displays no x label.
ylab	the y label of the plot. If missing, ylab = "Survival Probability" is used. Setting ylab = "" displays no y label.
lwd	the line width for the plot. If missing, lwd = 3 is used.
lty	the line type for the plot. If missing, lty = 1 is used.
plotCI	logical. If TRUE and x contains the estimates of the confidence intervals, these are plotted along with the survival function.
ltyCI	a graphical parameter defining the line type of the confidence interval. By default, ltyCI = 5 (dashed line).
...	optional arguments. Currently not in use.

See Also

turnbull.sb, turnbull.db, summary.turnbull

predict.dbchoice *Predicting model for dbchoice*

Description

Predicting method for objects of class "dbchoice".

Usage

```
## S3 method for class 'dbchoice'
predict(object, newdata = NULL, type = c("utility", "probability"),
bid = NULL, ...)
```

Arguments

object	an object of class "dbchoice".
newdata	a data frame containing new data to predict. If NULL, the original data is used.
type	type of prediction (utility or probability).
bid	a bid design needed to predict with original data.
...	optional arguments. Currently not in use.

Details

The function `predict()` for S3 object "dbchoice" calculates predicted values according to the fitted model that is included in `object`. The values are predicted with the original data used for fitting the model if `newdata = NULL`, otherwise with a new data assigned to `newdata`. There are two notes for `dbchoice()`: a bid design used for the fit must be assigned to `bid`, when predicting with the original data; the predicted values are calculated according to the relevant single-bounded dichotomous choice model, when predicting with a new data. See examples for details.

The current function does not estimate standard errors of predicted values.

Value

When `newdata = NULL` and `type = utility`, a matrix containing utility values under first (`f`), second upper (`u`), and second lower (`l`) bids is returned. When `newdata = NULL` and `type = probability`, a matrix containing probabilities of choosing Yes-Yes (`yy`), No-No (`nn`), Yes-No (`yn`), and No-Yes (`ny`) is returned. When a new data is assigned to `newdata`, predictions are calculated according to the relevant single-bounded dichotomous choice model, and a vector containing utility values of choosing yes (`type = utility`) or probability of choosing yes (`type = probability`) is returned.

See Also

dbchoice

Examples

```
## See Examples in dbchoice.
```

predict.sbchoice *Predicting model for sbchoice*

Description

Predicting method for objects of class "sbchoice".

Usage

```
## S3 method for class 'sbchoice'
predict(object, newdata = NULL, type = c("utility", "probability"),
...)
```

Arguments

`object`	an object of class "sbchoice".
`newdata`	a data frame containing new data to predict. If `NULL`, the original data is used.
`type`	type of prediction (utility or probability).
`...`	optional arguments. Currently not in use.

Details

The function `predict()` for S3 object "sbchoice" calculates predicted values according to the fitted model that is included in `object`. The values are predicted with the original data used for fitting the model if `newdata = NULL`, otherwise with a new data assigned to `newdata`.

The current function does not estimate standard errors of predicted values.

Value

When `newdata = NULL` and `type = utility`, a vector containing utility values of choosing Yes under bid values is returned. When `newdata = NULL` and `type = probability`, a vector containing probabilities of choosing Yes is returned. When a new data is assigned to `newdata`, a vector containing utility values of choosing Yes (`type = utility`) or probability of choosing Yes (`type = probability`) under the new data is returned.

See Also

`sbchoice`

Examples

```
## See Examples in sbchoice.
```

sbchoice	*Parametric approach to analyze single-bounded dichotomous choice contingent valuation data*

Description

This function analyzes single-bounded dichotomous choice contingent valuation (CV) data on the basis of the utility difference approach.

Usage

```
sbchoice(formula, data, subset, na.action = na.omit,
         dist = "log-logistic", ...)

## S3 method for class 'sbchoice'
print(x, digits = max(3, getOption("digits") - 1), ...)

## S3 method for class 'sbchoice'
vcov(object, ...)

## S3 method for class 'sbchoice'
logLik(object, ...)
```

Arguments

`formula`	an S3 class object "`formula`" and specifies the model structure.
`data`	a data frame containing the variables in the model formula.
`subset`	an optional vector specifying a subset of observations.
`na.action`	a function which indicates what should happen when the data contains `NA`s.
`dist`	a character string setting the error distribution in the model, which takes one of "`logistic`", "`normal`", "`log-logistic`", "`log-normal`" or "`weibull`".
`x`	an object of class "`sbchoice`".
`digits`	a number of digits to display.
`object`	an object of class "`dbchoice`".
`...`	optional arguments. Currently not in use.

Details

The function `sbchoice()` implements an analysis of single-bounded dichotomous choice contingent valuation (CV) data on the basis of the utility difference approach (Hanemann, 1984).

The extractor function `summary()` is available for a "`sbchoice`" class object. See `summary.sbchoice` for details.

There are two functions available for computing the confidence intervals for the estimates of WTPs. `krCI` implements simulations to construct empirical distributions of the WTP while `bootCI` carries out nonparametric bootstrapping.

Most of the details of `sbchoice()` is the same as those of `dbchoice()`, a double-bounded analogue of `sbchoice`. See the section **Details** in `dbchoice`. Differences between the two functions are as follows:

- In the model formula, the first part contains only one response variable (e.g., `R1`) and the third part contains only one bid variable (e.g., `BD1`) because respondents are requested to answer a CV question in the single-bounded dichotomous choice CV. The following is a typical structure of the formula:
  `R1 ~ (the names of the covariates) | BD1`
- The function `sbchoice()` analyzes the responses to single-bounded dichotomous choice CV questions internally using the function `glm()` with the argument
  `family = binomial(link = "logit")`
  or
  `family = binomial(link = "probit")`.
  When `dist = "weibull"`, optimization is carried out using the `optim()` function with a hard-coded log-likelihood function.
- Outputs from `sbchoice()` are slightly different from those from `dbchoice()` because the analysis in `sbchoice()` internally depends on the function `glm()` for the (log-) normal or (log-) logistic distributions. (see the **Value** section).

Nonparametric analysis of single-bounded dichotomous choice data can be done by `turnbull.sb` or by `kristrom`.

Value

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

`coefficients`	a named vector of estimated coefficients.
`call`	the matched call.
`formula`	the formula supplied.
`glm.out`	a list of components returned from `glm()` with the data set and the formula. In case of the Weibull distribution, a list of components from the `optim()` is saved.
`glm.null`	a list of components returned from `glm()` with the data set and a formula containing only constant (null model). In case of the Weibull distribution, a list of components from the `optim()` is saved.
`distribution`	a character string showing the error distribution used.
`nobs`	a number of observations.
`covariates`	a named matrix of the covariates used in the model.
`bid`	a named matrix of the bids used in the model.
`yn`	a named matrix of the responses to the CV question used in the model.
`data.name`	the data matrix.
`terms`	terms
`contrast`	contrasts used for factors
`xlevels`	levels used for factors

References

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

Boyle KJ, Welsh MP, Bishop RC (1988). "Validation of Empirical Measures of Welfare Change: Comment." *Land Economics*, **64**(1), 94–98.

Carson RT, Hanemann WM (2005). "Contingent Valuation." in KG M\"aler, JR Vincent (eds.), *Handbook of Environmental Economics*. Elsevier, New York.

Croissant Y (2011). **Ecdat***: Data Sets for Econometrics,* R package version 0.1-6.1, https://CRAN.R-project.org/package=Ecdat.

Hanemann, WM (1984). "Welfare Evaluations in Contingent Valuation Experiments with Discrete Responses", *American Journal of Agricultural Economics*, **66**(2), 332–341.

Hanemann M, Kanninen B (1999). "The Statistical Analysis of Discrete-Response CV Data.", in IJ Bateman, KG Willis (eds.), *Valuing Environmental Preferences: Theory and Practice of the Contingent Valuation Methods in the US, EU, and Developing Countries*, 302–441. Oxford University Press, New York.

See Also

summary.sbchoice, krCI, bootCI, NaturalPark, turnbull.sb, kristrom glm, formula dbchoice

Examples

```
## Examples for sbchoice() are also based on a data set NaturalPark
## in the package Ecdat (Croissant 2011): so see the section Examples
## in the dbchoice() for details.
data(NaturalPark, package = "Ecdat")

## The variable answers are converted into a format that is suitable for
## the function sbchoice() as follows:
NaturalPark$R1 <- ifelse(substr(NaturalPark$answers, 1, 1) == "y", 1, 0)
NaturalPark$R2 <- ifelse(substr(NaturalPark$answers, 2, 2) == "y", 1, 0)


## We assume that the error distribution in the model is a log-logistic;
## therefore, the bid variables bid1 is converted into LBD1 as follows:
NaturalPark$LBD1 <- log(NaturalPark$bid1)

## The utility difference function is assumed to contain covariates
## (sex, age, and income) as well as the bid variable (LBD1) as follows
## (R2 is not used because of single-bounded dichotomous choice CV format):
fmsb <- R1 ~ sex + age + income | LBD1

## Not run:
## The formula may be alternatively defined as
fmsb <- R1 ~ sex + age + income | log(bid1)

## End(Not run)
```

```
## The function sbchoice() with the function fmsb and the data frame NP
## is executed as follows:
NPsb <- sbchoice(fmsb, data = NaturalPark)
NPsb
NPsbs <- summary(NPsb)
NPsbs

## Not run:
## Generic functions such as summary() and coefficients() work for glm.out
summary(NPsb$glm.out)
coefficients(NPsb$glm.out)

## The confidence intervals for these WTPs are calculated using the
## function krCI() or bootCI() as follows:
krCI(NPsb)
bootCI(NPsb)
## The WTP of a female with age = 5 and income = 3 is calculated
## using function krCI() or bootCI() as follows:
krCI(NPsb, individual = data.frame(sex = "female", age = 5, income = 3))
bootCI(NPsb, individual = data.frame(sex = "female", age = 5, income = 3))

## End(Not run)

## The variable age and income are deleted from the fitted model,
## and the updated model is fitted as follows:
update(NPsb, .~. - age - income |.)

## Respondents' utility and probability of choosing Yes under
## the fitted model and original data are predicted as follows:
head(predict(NPsb, type = "utility"))
head(predict(NPsb, type = "probability"))
## Utility and probability of choosing Yes for a female with age = 5
## and income = 3 under bid = 10 are predicted as follows:
predict(NPsb, type = "utility",
    newdata = data.frame(sex = "female", age = 5, income = 3, LBD1 = log(10)))
predict(NPsb, type = "probability",
    newdata = data.frame(sex = "female", age = 5, income = 3, LBD1 = log(10)))

## Plot of probabilities of choosing yes is drawn as drawn as follows:
plot(NPsb)
## The range of bid can be limited (e.g., [log(10), log(20)]):
plot(NPsb, bid = c(log(10), log(20)))
```

`summary.dbchoice` *Summarizing* `dbchoice` *estimation*

Description

Summary method for objects of class "`dbchoice`".

Usage

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

## S3 method for class 'summary.dbchoice'
print(x, digits = max(3, getOption("digits") - 1), ...)
```

Arguments

`object`	an object of class "dbchoice".
`x`	an object of class "summary.dbchoice".
`digits`	a number of digits to display.
`...`	optional arguments. Currently not in use.

Details

The function `summary.dbchoice()` computes and returns a list of summary statistics of the fitted model in `object` of the "dbchoice" class.

Some of the values are printed up to certain decimal places. Actual values of individual components are displayed separately, for instance, by `summary(object)$coefficients`. See the **Value** section for a list of components.

Value

In addition to those available in the `object` of the "dbchoice" class, the following list components are added.

`medianWTP`	the estimated median WTP.
`meanWTP`	the estimated mean WTP.
`trunc.meanWTP`	the estimated mean WTP truncated at the maximum bid.
`adj.trunc.meanWTP`	the truncated mean WTP with the adjustment of Boyle *et~al.*(1988).
`LR.test`	a vector containing the likelihood ratio test statistic, the degrees of freedom and the associated p-value. The null is that all the parameters on the explanatory variables other than constant and the bid variable are jointly zero.
`coef`	a table of estimates including their s.e., z-value, and p-value.
`AIC`	information criterion (AIC and BIC).

References

Boyle KJ, Welsh MP, Bishop RC (1988). "Validation of Empirical Measures of Welfare Change: Comment." *Land Economics*, **64**(1), 94–98.

See Also

`dbchoice`

summary.kristrom *Summarizing the Kristr\"om's nonparametric estimation of WTP*

Description

Summary method for objects of class "kristrom".

Usage

```
## S3 method for class 'kristrom'
summary(object, digits = max(3, getOption("digits") - 1), ...)

## S3 method for class 'summary.kristrom'
print(x, digits = max(3, getOption("digits") - 1), ...)
```

Arguments

object	an object of class "kristrom".
x	an object of class "kristrom".
digits	a number of digits to display.
...	optional arguments. Currently not in use.

See Also

plot.kristrom, kristrom, NaturalPark, sbchoice

summary.sbchoice *Summarizing* sbchoice *estimation*

Description

Summary method for objects of class sbchoice.

Usage

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

## S3 method for class 'summary.sbchoice'
print(x, digits = max(3, getOption("digits") - 1), ...)
```

Arguments

`object`	an object of class `"sbchoice"`.
`x`	an object of class `"summary.sbchoice"`.
`digits`	a number of digits to display.
`...`	optional arguments. Currently not in use.

Details

The function `summary.sbchoice()` computes and returns a list of summary statistics of the fitted model in `object` of the `"sbchoice"` class.

Some of the values are printed up to certain decimal places. Actual values of individual components are displayed separately, for instance, by `summary(object)$coefficients`. See the **Value** section for a list of components.

Since the model for the single-bounded dichotomous choice CV data is estimated by `glm`, an object of class
`"summary.sbchoice"` is constructed based on a `"summary.glm"` class object. The summary of the `"summary.glm"` class object is available by `summary(object)$glm.summary`. Other components computed by `summary.glm` are also accessible. See `summary.glm` for details.

Value

In addition to those available in the `object` of the `"sbchoice"` class, the following list components are added.

`glm.summary`	a summary of the `glm` estimation. It is an object of class `"summary.glm"`.
`glm.null.summary`	a summary of the `glm` estimation of the null model (i.e., only with the intercept). It is an object of class `"summary.glm"`.
`loglik`	the value of the log-likelihood of the model.
`loglik.null`	the value of the log-likelihood of the null model.
`psdR2`	McFadden's pseudo-R2 measure.
`adjpsdR2`	McFadden's pseudo-R2 measure adjusted for the degrees of freedom.
`medianWTP`	the estimated median WTP.
`meanWTP`	the estimated mean WTP.
`trunc.meanWTP`	the estimated mean WTP truncated at the maximum bid.
`adj.trunc.meanWTP`	the truncated mean WTP with the adjustment of Boyle *et~al.*(1988).
`LR.test`	a vector containing the likelihood ratio test statistic, the degrees of freedom and the associated p-value.
`AIC`	information criterion (AIC and BIC).

References

Boyle KJ, Welsh MP, Bishop RC (1988). "Validation of Empirical Measures of Welfare Change: Comment." *Land Economics*, **64**(1), 94–98.

See Also

sbchoice, glm, summary.glm

---

summary.turnbull *Summarizing the Kaplan-Meier-Turnbull nonparametric estimation of WTP*

Description

Summary method for objects of class "turnbull".

Usage

```
## S3 method for class 'turnbull'
summary(object, printCI = TRUE, ...)

## S3 method for class 'summary.turnbull'
print(x, digits = max(3, getOption("digits") - 1), ...)
```

Arguments

object	an object of class "turnbull".
printCI	an argument controlling whether the summary of confidence interval estimates are printed. The CIs are not summarized and printed unless conf.int = TRUE in turnbull.sb or turnbull.db.
x	an object of class "turnbull".
digits	a number of digits to display.
...	optional arguments. Currently not in use.

---

turnbull *The Kaplan-Meier-Turnbull nonparametric approach to analyze dichotomous choice contingent valuation data*

Description

This function analyzes single-, one-and-one-half-, or double-bounded dichotomous choice contingent valuation (CV) data on the basis of the Kaplan-Meier-Turnbull method.

Usage

```
## for the single-bounded data
turnbull.sb(formula, data, subset, conf.int = FALSE, B = 200,
            conf.level = 0.95, timeMessage = FALSE, seed = 19439101, ...)

## for the one-and-one-half-bound data
turnbull.oohb(formula, data, subset, conf.int = FALSE, B = 200,
              conf.level = 0.95, timeMessage = FALSE, seed = 19439101, ...)


## for the double-bounded data
turnbull.db(formula, data, subset, conf.int = FALSE, B = 200,
            conf.level = 0.95, timeMessage = FALSE, seed = 19439101, ...)

## S3 method for class 'turnbull'
print(x, digits = max(3, getOption("digits") - 1), ...)
```

Arguments

`formula`	a formula specifying the model structure.
`data`	a data frame containing the variables in the model formula.
`subset`	an optional vector specifying a subset of observations.
`x`	an object of class `"turnbull"`.
`conf.int`	logical. If `TRUE`, the confidence intervals are computed.
`B`	number of bootstrap resampling for confidence interval estimation. Defaulted to 200.
`conf.level`	a confidence coefficient. Defaulted to 0.95.
`timeMessage`	logical. If `TRUE`, the estimated time for bootstrapping is displayed.
`digits`	a number of digits to display.
`seed`	a seed for a random number generator.
`...`	optional arguments. Currently not in use.

Details

The functions `turnbull.sb()`, `turnbull.oohb()`, and `turnbull.db()` analyze single-, one-and-one-half-, and double-bounded dichotomous choice contingent valuation (CV) data, respectively, on the basis of the modified Kaplan-Meier-Turnbull method (Carson and Steinberg, 1990).

For single-bounded dichotomous choice data
Most of the details of `turnbull.sb()` is the same as those of `turnbull.db()`. See the subsequent section for details.

A difference between `turnbull.sb()` and `turnbull.db()` appears in the definition of the model formula. In `turnbull.sb()`, the first part contains only one response variable (e.g., `R1`) and the other part contains only one bid variable (e.g., `BD1`) because respondents are requested to answer a CV question in the single-bounded dichotomous choice CV. A typical structure of the formula is given by

R1 ~ BD1

kristrom is an alternative nonparametric method for analyzing single-bounded dichotomous choice data. A parametric analysis can be done by sbchoice.

For one-and-one-half-bound dichotomous choice data
The details of turnbull.oohb() is the same as those of turnbull.db(). See the the subsequent section for details.

A difference between turnbull.oohb() and turnbull.db() appears in the definition of variables. See oohbchoice and oohbsyn for the details of creating response and bid variables.

For double-bounded dichotomous choice data
A generic call to turnbull.db() is given by

turnbull.db(formula,data)

The argument formula defines the response variables and bid variables. The argument data is set as a data frame containing the variables in the model.

A typical structure of the formula for turnbull.db() is defined as follows:

R1 + R2 ~ BD1 + BD2

The formula consists of two parts. The first part, the left-hand side of the tilde sign (~), must contain the response variables for the suggested prices in the first and the second stage of CV questions. In the above example, R1 denotes a binary or two-level factor response variable for a bid in the first stage and R2 for a bid in the second stage. Each of R1 and R2 contains "Yes" or "No" to the bid or 1 for "Yes" and 0 for "No". The two variables are connected with the arithmetic operator (+). The other part, which starts after the tilde sign, must contain bid variables (BD1, BD2) containing suggested prices in the first and second stage of double-bounded dichotomous choice CV question. The two variables are also connected with the arithmetic operator (+).

A parametric approach for analyzing double-bounded dichotomous choice data can be carried out by dbchoice.

The structure of data set which assigned to the argument data is the same as that of dbchoice(). See dbchoice for details in the data set structure.

The confidence intervals are computed by the bootstrap method in icfit of the **interval** (Fay and Shaw, 2010) package. The arguments conf.int, B, conf.level and timeMessage are passed to icfit(). The bootstrap can be time consuming, so that it is in general not advisable to increase the number of bootstrap resamplint B. See the help of icfit for futher detail.

The generic function print() is available for a fitted model object of class "turnbull" and displays simply estimated probabilities of the distribution: this is the same as the element pf in the function icfit of the **interval** (Fay and Shaw, 2010) package.

The extractor function summary() is also available for a fitted model object of class "turnbull" and the generic function print.summary() displays the survival probability and three types of WTP estimates (a Kaplan-Meier mean, a Spearman-Karber mean, and median WTPs). In estimating the two types of mean WTP, the area under the survivor function is truncated at the maximum bid because there seems no unified approach to determine an ending point of bids at which the acceptance probability is zero. Therefore, we leave the decision of how the area is treated to the user. In practice, once the ending point, bid_end, is found, it is straightforward to compute the triangular area by $0.5(bid_end - bid_max)P_max$ where bid_max is the maximum bid and P_max is the acceptance probability for bid_max, both of which are reported in the summarized output.

Furthermore, the generic function plot() is available for a fitted model object of class "turnbull" and displays the empirical survival function. See plot.turnbull for details.

Value

Both turnbull.db() and turnbull.sb() return an object of S3 class "turnbull" that is a list with the following components.

left	a vector of left endpoints of censored interval. The vector is internally assigned to the argument L in icfit() of the **interval** package.
right	a vector of right endpoints of censored interval. The vector is internally assigned to the argument R in icfit() of the **interval** package.
turnbull	a list of components returned from icfit().
unq.dib	a vector of unique bids including Inf.
CI	estimates for confidence intervals from icfit().

References

Croissant Y (2011). **Ecdat***: Data Sets for Econometrics,* R package version 0.1-6.1, https://CRAN.R-project.org/package=Ecdat.

Fay MP, Shaw PA (2010). "Exact and Asymptotic Weighted Logrank Tests for Interval Censored Data: The **interval** R Package", *Journal of Statistical Software*, **36**(2), 1-34. https://www.jstatsoft.org/v36/i02/.

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

Krinsky I, Robb AL (1990). "On Approximating the Statistical Properties of Elasticities: A Correction." *The Review of Economics and Statistics*, **72**, 189–190.

Turnbull BW (1976). "The Empirical Distribution Function with Arbitrarily Grouped, Censored and Truncated Data." *Journal of the Royal Statistical Society. Series B*, **38**(3), 290–295.

See Also

summary.turnbull, plot.turnbull, kristrom, sbchoice, dbchoice, NaturalPark, glm, icfit

Examples

```
## Examples are based on turnbull.db(). turnbull.sb() is similarly
## implemented. A difference appears in the definition of the model
## formula. See "Details" section of the help.

## A data set used here is NaturalPark in the package Ecdat (Croissant,
## 2011): double-bounded dichotomous choice CV style question for
## measuring willingness to pay for the preservation of the Alentejo
## Natural Park. The data frame contains seven variables: bid1 (bid in
## the initial question), bidh (higher bid in the follow-up question),
## bidl (lower bid in the follow-up question), answers (response
## outcomes in a factor format with four levels of "nn", "ny", "yn", "yy"),
## respondents' characteristic variables such as age, sex and income (see
## NaturalPark for details).
data(NaturalPark, package = "Ecdat")

## The variable answers are converted into a format that is suitable for
```

```
## the function turnbull.db() as follows:
NaturalPark$R1 <- ifelse(substr(NaturalPark$answers, 1, 1) == "y", 1, 0)
NaturalPark$R2 <- ifelse(substr(NaturalPark$answers, 2, 2) == "y", 1, 0)

## The variables bidh and bidl are integrated into one variable (bid2)
## as follows:
NaturalPark$bid2 <- ifelse(NaturalPark$R1 == 1, NaturalPark$bidh, NaturalPark$bidl)

## The formula for turnbull.sb and turnbull.db are defined respectively as follows:
fmts <- R1 ~ bid1
fmtd <- R1 + R2 ~ bid1 + bid2

## The function turnbull.db() with the formula fmtd and the data frame
## NaturalPark is executed as follows:
NPts <- turnbull.sb(fmts, data = NaturalPark)
NPts
NPtss <- summary(NPts)
NPtss
plot(NPts)


## The function turnbull.db() with the formula fmtd and the data frame
## NaturalPark is executed as follows:
NPtd <- turnbull.db(fmtd, data = NaturalPark)
NPtd
NPtds <- summary(NPtd)
NPtds
plot(NPtd)
```

---

update.dbchoice *Updating and refitting model for dbchoice*

Description

Updating and refitting method for object of class "dbchoice".

Usage

```
## S3 method for class 'dbchoice'
update(object, new, evaluate = TRUE, ...)
```

Arguments

object	an object of class "dbchoice".
new	a new call.
evaluate	If TRUE, refit the updated model, otherwise return the updated call.
...	optional arguments. Currently not in use.

Details

The function `update()` for S3 object "dbchoice" updates a model used for the fit that is included in `object` according to a new call assigned to `new`, and then refits the updated model. The function returns the refitted model object when `evaluate = TRUE`, otherwise the updated model call.

See Also

`dbchoice`

Examples

```
## See Examples in dbchoice.
```

`update.sbchoice` *Updating and refitting model for sbchoice*

Description

Updating and refitting method for object of class "sbchoice".

Usage

```
## S3 method for class 'sbchoice'
update(object, new, evaluate = TRUE, ...)
```

Arguments

`object`	an object of class "sbchoice".
`new`	a new call.
`evaluate`	If `TRUE`, refit the updated model, otherwise return the updated call.
`...`	optional arguments. Currently not in use.

Details

The function `update()` for S3 object "sbchoice" updates a model used for the fit that is included in `object` according to a new call assigned to `new`, and then refits the updated model. The function returns the refitted model object when `evaluate = TRUE`, otherwise the updated model call.

See Also

`sbchoice`

Examples

```
## See Examples in sbchoice.
```

Index