Package 'cjoint'

July 22, 2025

Version 2.1.1

Date 2023-08-21

Title AMCE Estimator for Conjoint Experiments

Author Soubhik Barari, Elissa Berwick, Jens Hainmueller, Daniel Hopkins, Sean Liu, Anton Strezhnev, Teppei Yamamoto

Maintainer Anton Strezhnev <astrezhnev@uchicago.edu>

Depends R (>= 2.10), sandwich, lmtest, ggplot2, survey, Matrix

Imports stats, utils, methods, DT, shiny, shinyjs

Suggests testthat

Description An R implementation of the Average Marginal Component-specific Effects (AMCE) estimator presented in Hainmueller, J., Hopkins, D., and Yamamoto T. (2014) <DOI:10.1093/pan/mpt024> Causal Inference in Conjoint Analysis: Understanding Multi-Dimensional Choices via Stated Preference Experiments. Political Analysis 22(1):1-30.

License GPL (>= 2)

RoxygenNote 6.1.0

Encoding UTF-8

NeedsCompilation no

Repository CRAN

Date/Publication 2023-08-22 07:50:07 UTC

Contents

`cjoint-package` *cjoint: A Package for Estimating Average Marginal Component-specific Effects for Conjoint Survey Experiments*

Description

This package allows researchers to estimate the causal effects of attributes in conjoint survey experiments. It implements the Average Marginal Component-specific Effects (AMCE) estimator presented in Hainmueller, J., Hopkins, D., and Yamamoto T. (2014) Causal Inference in Conjoint Analysis: Understanding Multi-Dimensional Choices via Stated Preference Experiments. Political Analysis 22(1):1-30

Details

Package:	cjoint
Type:	Package
Version:	2.1.1
Date:	2023-08-21
License:	GPL (>= 2)

Author(s)

Authors: Soubhik Barari, Elissa Berwick, Jens Hainmueller, Daniel Hopkins, Sean Liu, Anton Strezhnev, Teppei Yamamoto

Maintainer: Anton Strezhnev <astrezhnev@uchicago.edu>

References

Hainmueller, J., Hopkins, D., and Yamamoto T. (2014) Causal Inference in Conjoint Analysis: Understanding Multi-Dimensional Choices via Stated Preference Experiments. Political Analysis 22(1):1-30

amce *Estimating Causal Effects in Conjoint Experiments*

Description

This function takes a dataset and a conjoint design and returns Average Marginal Component Effects (AMCEs) and Average Component Interaction Effects (ACIE) for the attributes specified in the formula. By default, this function assumes uniform randomization of attribute levels and no profile restrictions. If your design incorporates weighted randomization or restrictions on displayable profiles, first generate a design object using `makeDesign`. Interactions with respondent-level characteristics are handled by identifying relevant variables as respondent-varying.

Usage

```
amce(formula, data, design = "uniform",
      respondent.varying = NULL, subset = NULL,
      respondent.id = NULL, cluster = TRUE, na.ignore=FALSE,
      weights = NULL, baselines = NULL)
```

Arguments

`formula`	A `formula` object specifying the name of the outcome variable on the left-hand side and the attributes for which effects are to be estimated on the right-hand side. RHS attributes should be separated by + signs. Interaction effects can be specified using standard interaction syntax - joining attribute names using either : or *. However using the : syntax will produce the same results as * since missing base terms are automatically added to the formula. For example `Y ~ X1 + X2` will return AMCEs for X1 and X2. `Y ~ X1 + X2 + X1:X2` will return AMCEs for X1 and X2 along with an ACIE for X1/X2. `Y ~ X1*X2` and `Y ~ X1:X2` will produce identical results to `Y ~ X1 + X2 + X1:X2`. Note that you can place backticks around a variable name containing spaces in order to have `formula` interpret it as a single variable name. Any respondent characteristics must be designated as such in `redpondent.varying`.
`data`	A dataframe containing the outcome variable, attributes, respondent identifiers, respondent covariate data and sampling weights from a conjoint experiment.
`design`	Either the character string `"uniform"` or a `conjointDesign` object created by the `makeDesign` function. If a `conjointDesign` is not passed, the function will assume all attribute levels have an equal probability of being presented to a respondent and that no profiles are restricted. Defaults to `"uniform"`.
`respondent.varying`	A vector of character strings giving the names of any respondent-varying characteristics being interacted with AMCEs or ACIEs in the `formula`.
`subset`	A logical vector with length `nrow(data)` denoting which rows in `data` should be included in estimation. This can for example be used to subset the data along respondent-level covariates. Defaults to `NULL`.

`respondent.id` A character string indicating the column of `data` containing a unique identifier for each respondent. Defaults to `NULL`.

`cluster` A logical indicating whether estimated standard errors should be clustered on `respondent.id`. Defaults to `TRUE`.

`na.ignore` A logical indicating whether the function should ignore missing rows in `data`. If `FALSE`, amce() will raise an error if there are rows with missing values. Defaults to `FALSE`.

`weights` A character string giving the name of the column in the data containing any survey weights. See documentation for `survey` package for more information.

`baselines` Manually adjust the baselines of select factor variables (either attributes or respondent varying) by supplying a list. Names of list entries should correspond with variable names. The content of each entry should be a character string giving the new baseline.

Value

An object of class "amce" containing:

`attributes` A list containing the names of attributes.

`baselines` Baseline levels for each attribute in `estimates`. Baselines determined using the first element of `levels()`. If a different baseline level is desired for an attribute, use the `relevel()` function on the variable prior to calling the `amce()` routine or supply an alternative baseline in `baselines` argument.

`continuous` List of quantiles for any non-factor variables, whether attributes or respondent varying.

`data` The original data.

`estimates` A list containing AMCE and ACIE estimates for each attribute in `formula`. Each element of `estimates` corresponds to a single attribute or interaction.

`formula` The `formula` passed to the `amce()` routine.

`samplesize_prof` The number of valid profiles (rows) in the dataset

`user.names` A vector with the original user supplied names for any attributes. These may differ from the attribute names in `estimates` if the original names contain spaces.

`vcov.prof` The modified variance-covariance matrix for AMCE and ACIE estimates. Incorporates cluster corrections as well as attribute dependencies. Profile varying attributes only.

`numrespondents` The number of respondents in the dataset (if `respondent.id` is not `NULL`).

`respondent.varying` Names of respondent-varying variables, if any.

`cond.formula` The formula used for calculating estimates conditional on respondent varying characteristics. Only returned when respondent-varying characteristics are present.

`cond.estimates` A list containing estimated effects of respondent-varying characteristics conditional on attribute values. Each element of `cond.estimates` corresponds to a single attribute or interaction. Only returned when respondent-varying characteristics are present. To obtain AMCE and ACIE estimates conditional on the values of the respondent-varying characteristics see `summary.amce`

`samplesize_full`
: The number of valid profiles (rows) in the dataset when respondent varying characteristics are included. Only returned when respondent-varying characteristics are present.

`vcov.resp`
: The modified variance-covariance matrix for effect estimates conditional on respondent-varying characteristics, where dependent relationships have been incorporated into variances and covariances. Only returned when respondent-varying characteristics are present.

References

Hainmueller, J., Hopkins, D., and Yamamoto T. (2014) Causal Inference in Conjoint Analysis: Understanding Multi-Dimensional Choices via Stated Preference Experiments. Political Analysis 22(1):1-30

See Also

`summary.amce` for summaries and `plot.amce` for generating a coefficient plot using `ggplot2`.

`makeDesign` to create `conjointDesign` objects.

Examples

```
## Not run:
# Immigration Choice Conjoint Experiment Data from Hainmueller et. al. (2014).
data("immigrationconjoint")
data("immigrationdesign")

# Run AMCE estimator using all attributes in the design
results <- amce(Chosen_Immigrant ~  Gender + Education + `Language Skills` +
                `Country of Origin` + Job + `Job Experience` + `Job Plans` +
                `Reason for Application` + `Prior Entry`, data=immigrationconjoint,
                cluster=TRUE, respondent.id="CaseID", design=immigrationdesign)
# Print summary
summary(results)


# Run AMCE estimator using all attributes in the design with interactions
interaction_results <- amce(Chosen_Immigrant ~  Gender + Education + `Language Skills` +
                `Country of Origin` + Job + `Job Experience` + `Job Plans` +
                `Reason for Application` + `Prior Entry` + Education:`Language Skills` +
Job: `Job Experience` + `Job Plans`:`Reason for Application`,
data=immigrationconjoint, cluster=TRUE, respondent.id="CaseID",
design=immigrationdesign)
# Print summary
summary(interaction_results)

# create weights in data
weights <- runif(nrow(immigrationconjoint))
immigrationconjoint$weights <- weights
# Run AMCE estimator using weights
results <- amce(Chosen_Immigrant ~  Gender + Education + `Language Skills` +
```

```
                `Country of Origin` + Job + `Job Experience` + `Job Plans` +
                `Reason for Application` + `Prior Entry`, data=immigrationconjoint,
                cluster=TRUE, respondent.id="CaseID", design=immigrationdesign,
weights = "weights")
# Print summary
summary(results)

# Include a respondent-varying interaction
results <- amce(Chosen_Immigrant ~ Gender + Education + Job +
    ethnocentrism:Job + Education:Job,
data=immigrationconjoint, na.ignore = TRUE,
cluster=FALSE,design=immigrationdesign,
respondent.varying = "ethnocentrism")
# Print summary
summary(results)

# Change the baseline for "Education"
baselines <- list()
baselines$Education <- "graduate degree"

results <- amce(Chosen_Immigrant ~ Gender + Education + Job +
 Education:Job, data=immigrationconjoint,
 cluster=FALSE,design=immigrationdesign,
 baselines=baselines)
# Print summary
summary(results)

## End(Not run)
```

`immigrationconjoint` *Immigration Conjoint Experiment Dataset from Hainmueller et. al. (2014)*

Description

A dataset containing the results of a conjoint survey of a representative sample of American adults who were asked to choose which hypothetical immigrants they think should be admitted into the United States. Each row corresponds to a single profile presented to the respondent.

Usage

```
data("immigrationconjoint")
```

Format

A data frame with 13,960 observations on the following 16 variables.

`CaseID` a numeric vector indicating the respondent to which the particular profile corresponds

`contest_no` a numeric vector indicating the number of the task to which the profile corresponds

`Education` a factor with levels `no formal`, `4th grade`, `8th grade`, `high school`, `two-year college`, `college degree`, `graduate degree`

`Gender` a factor with levels `female`, `male`

'`Country of Origin`' a factor with levels `India`, `Germany`, `France`, `Mexico`, `Philippines`, `Poland`, `China`, `Sudan`, `Somalia`, `Iraq`

'`Reason for Application`' a factor with levels `reunite with family`, `seek better job`, `escape persecution`

`Job` a factor with levels `janitor`, `waiter`, `child care provider`, `gardener`, `financial analyst`, `construction worker`, `teacher`, `computer programmer`, `nurse`, `research scientist`, `doctor`

'`Job Experience`' a factor with levels `none`, `1-2 years`, `3-5 years`, `5+ years`

'`Job Plans`' a factor with levels `will look for work`, `contract with employer`, `interviews with employer`, `no plans to look for work`

'`Prior Entry`' a factor with levels `never`, `once as tourist`, `many times as tourist`, `six months with family`, `once w/o authorization`

'`Language Skills`' a factor with levels `fluent English`, `broken English`, `tried English but unable`, `used interpreter`

`Chosen_Immigrant` a numeric vector denoting whether the immigrant profile was selected

`ethnocentrism` a numeric vector

`profile` a numeric vector giving the profile number

`LangPos` a numeric vector

`PriorPos` a numeric vector

Source

Hainmueller, J., Hopkins, D., and Yamamoto T. (2014) Causal Inference in Conjoint Analysis: Understanding Multi-Dimensional Choices via Stated Preference Experiments. Political Analysis 22(1):1-30

Examples

```
## Not run:
data("immigrationconjoint")
data("immigrationdesign")

# Run AMCE estimator using all attributes in the design
results <- amce(Chosen_Immigrant ~  Gender + Education + `Language Skills` +
                `Country of Origin` + Job + `Job Experience` + `Job Plans` +
                `Reason for Application` + `Prior Entry`, data=immigrationconjoint,
                cluster=TRUE, respondent.id="CaseID", design=immigrationdesign)

## End(Not run)
```

`immigrationdesign` *Conjoint Design for the Immigration Experiment in Hainmueller et. al. (2014)*

Description

A "conjointDesign" object for the randomization scheme used in the immigration conjoint experiment in "Causal Inference in Conjoint Analysis." See `immigrationconjoint` for the accompanying dataset.

Usage

```
data("immigrationdesign")
```

Format

A "conjointDesign" object. See `makeDesign` for more information about the structure.

Source

Hainmueller, J., Hopkins, D., and Yamamoto T. (2014) Causal Inference in Conjoint Analysis: Understanding Multi-Dimensional Choices via Stated Preference Experiments. Political Analysis 22(1):1-30

Examples

```
## Not run:
# Loads the immigrationconjoint dataset and the immigrationdesign object
data("immigrationconjoint")
data("immigrationdesign")

# immigrationdesign is passed to the amce() function through the "design" argument.
results <- amce(Chosen_Immigrant ~  Gender + Education + `Language Skills` +
                `Country of Origin` + Job + `Job Experience` + `Job Plans` +
                `Reason for Application` + `Prior Entry`, data=immigrationconjoint,
                cluster=TRUE, respondent.id="CaseID", design=immigrationdesign)


## End(Not run)
```

japan2014conjoint *Japan 2014 Conjoint Experiment Dataset from Horiuchi et. al. (2014)*

Description

A dataset containing the results of a conjoint survey of a representative sample of Japanese adults who were asked to choose which of two parties (with specified policy manifestos) they would support in the 2014 House of Representatives general election. Each row corresponds to a single profile presented to the respondent. See the original replication archive (Horiuchi et. al. 2014) for more detailed descriptions.

Usage

```
data("japan2014conjoint")
```

Format

A data frame with 20,360 observations on the following 11 variables.

respondent a character vector uniquely identifying the respondent

respondentIndex a numeric vector uniquely indexing the respondent

task a numeric vector indexing the task presented to the respondent

profile a numeric vector indexing the profile presented to the respondent

selected a numeric vector indicating whether the profile was selected

Consumption tax a factor indicating the profile position on 'Consumption Tax'

Consumption tax.rowpos a numeric vector indicating which row the attribute 'Consumption Tax' appeared in the given profile

Employment a factor indicating the profile position on 'Employment'

Employment.rowpos a numeric vector indicating which row the attribute 'Employment' appeared in the given profile

Monetary and fiscal policy a factor indicating the profile position on 'Monetary and Fiscal Policy'

Monetary and fiscal policy.rowpos a numeric vector indicating which row the attribute 'Monetary and Fiscal Policy' appeared in the given profile

Economic growth strategy a factor indicating the profile position on 'Economic Growth Strategy'

Economic growth strategy.rowpos a factor indicating which row the attribute 'Economic Growth Strategy' appeared in the given profile

Nuclear power a factor indicating the profile position on 'Nuclear Power'

Nuclear power.rowpos a factor indicating which row the attribute 'Nuclear Power' appeared in the given profile

TPP a factor indicating the profile position on 'Trans-Pacific Partnership (TPP)'

TPP.rowpos a factor indicating which row the attribute 'Trans-Pacific Partnership (TPP)' appeared in the given profile

Collective self-defense a factor indicating which row the attribute 'Collective Self-Defense' appeared in the given profile

Collective self-defense.rowpos a factor indicating which row the attribute 'Collective Self-Defense' appeared in the given profile

Constitutional revision a factor indicating the profile position on 'Constitutional Revision'

Constitutional revision.rowpos a factor indicating which row the attribute 'Constitutional Revision' appeared in the given profile

National assembly seat reduction a factor indicating the profile position on 'National Assembly Seat Reduction'

National assembly seat reduction.rowpos a factor indicating which row the attribute 'National Assembly Seat Reduction' appeared in the given profile

wgt post-stratification weights to map the survey sample to the census population

Note

The origional survey was conducted in Japanese. To comply with CRAN's policy, we translate the variable names into English.

Source

Horiuchi, Yusaku; Smith, Daniel M.; Yamamoto, Teppei, 2017, "Replication Data for: Measuring Voters' Multidimensional Policy Preferences with Conjoint Analysis: Application to Japan's 2014 Election", https://doi.org/10.7910/DVN/KUMMUJ, Harvard Dataverse, V1

Examples

```
data("japan2014conjoint")

# Run AMCE estimator using all attributes and uniform design
results <- amce(selected ~ `Consumption tax` + `Employment` + `Monetary and fiscal policy` +
                  `Economic growth strategy` + `Nuclear power` + `TPP` +
                  `Collective self-defense` + `Constitutional revision` +
               `National assembly seat reduction`,  data=japan2014conjoint, cluster=TRUE,
                  respondent.id="respondentIndex", weights="wgt", design="uniform")
```

---

makeDesign *Create conjoint design*

Description

Generates a "conjointDesign" object to be passed to amce. Average Marginal Component Effects (AMCEs) are defined relative to the distribution of potential choice profiles. When the probability of each profile being presented to a respondent is not constant (e.g. some profiles are restricted from appearing), some attributes will not be independent and simple difference-in-means estimates will not be unbiased for the AMCE. This function allows users to specify non-uniform profile assignment schemes to be used by amce.

Usage

```
makeDesign(type="file", J=NULL, filename=NULL, attribute.levels = NULL,
           constraints=NULL, level.probs=NULL, tol=1e-14)
```

Arguments

type	A character string - either "file", "constraints", or "array". Each option requires a different set of additional arguments. If "file", the user must specify a filename of a ".dat" file exported via the Conjoint Survey Design Tool. If "constraints," the user must provide a list of attributes and levels in "attribute.levels" along with any constraints in "constraints" and attribute randomization weights in "level.probs". If "array," the user must pass an array to J.
J	If type="array", makeDesign requires a d-dimensional array to be passed to J where d is the number of attributes in a single profile. . Each dimension should have a number of indices corresponding to the number of levels for that attribute. Attribute and level names are taken from the dimnames of the array (level names are taken from the list elements, attribute names are taken from the names of the elements). Each cell of the array is the joint probability of assigning that particular profile (combination of attribute-levels).
filename	If type="file", this is a character string giving the name of a design file exported from the Conjoint Survey Design Tool using the "Export design to R" option.
attribute.levels	If type="constraints", attribute.levels is a required argument. This takes a named list with each element containing a character vector with the level names of a single attribute. The names of the elements should be the attribute names.
constraints	If type = "constraints", this is an optional argument. A list of lists. Each element list is a "restriction" and describes a set of restricted profiles (attribute-level groupings that cannot be displayed to a respondent) with character vectors containing the attribute-levels that cannot be displayed together. If your design prevents certain profiles from being seen by respondents (for example, if the particular profile would be highly impluasible), this argument allows you to specify those restrictions. Each element of a restriction is a character vector of levels for a particular attribute. That element must have the name of one of the attributes. For each specified restriction, makeDesign will treat any profile containing any combination of the restricted levels as having zero probability of being displayed. For example, if a restriction contains two elements: A = c("1","2") and B = c("4", "5"), then any profiles containing levels 1 or 2 of attribute A and levels 4 or 5 of attribute B will restricted from the design.
level.probs	If type = "constraints", this is an optional argument. A named list containing numeric vector elements. Each list element should have the name of one of the attributes passed in attribute.levels and the numeric vector it contains should have the names of the corresponding levels. Each element of the numeric vector specifies the marginal probability of that level appearing in a profile before any restricted profiles are eliminated. Each of the vectors should sum to 1. If NULL, then uniform randomization is assumed (excluding constrained profiles).
tol	A very small non-zero number used in the routine for determining which attributes are dependent (the distribution of one attribute depends on the value

of another attribute). Two attributes are independent if the distribution of one attribute does not change conditional on the other - this is calculated automatically through the J matrix Do not change unless you are having issues with `makeDesign` incorrectly labeling independent attributes as dependent (in which case, lower the value) or incorrectly labeling dependent attributes as independent (in which case, raise the value).

Details

See Hainmueller et. al. (2014) for details on the AMCE estimator in the presence of conditionally independent attributes. If you have no restricted profiles in your design and uniform randomization across profiles is used, then creating a design object is not necessary (the regression coefficients will give you AMCEs). However, if some profile combinations are restricted or some profiles are made to appear more often than others, it is necessary to specify them using `makeDesign` in order to obtain the correct AMCE estimates using `amce`.

You can create designs with assignment probabilities different from those of your original experiment. This allows you to estimate AMCEs for another tailored user-defined distribution of attribute-levels, such as the empirical distribution of the attributes in the population of interest. Note that the distribution of the original experiment must have a support that covers the entire support of the custom-made distribution. See Hainmueller et. al. (2014) for more details about the AMCE estimator and its relationship to the attribute-level distribution.

Value

Returns an object of class "conjointDesign" containing:

`J` A d-dimensional array where d is the number of attributes in a single profile. Each dimension has a number of indices corresponding to the number of levels for that attribute. Each entry is the probability of that profile being displayed to a survey respondent. The `dimnames` of the array contain the attribute and level names.

`dependence` A list of character vectors denoting which attributes are marginally dependent - used in the `amce` estimation routine.

References

Strezhnev, A., Hainmueller, J., Hopkins, D., and Yamamoto, T. (2014) Conjoint Survey Design Tool. https://github.com/astrezhnev/conjointsdt/

Hainmueller, J., Hopkins, D., and Yamamoto T. (2014) Causal Inference in Conjoint Analysis: Understanding Multi-Dimensional Choices via Stated Preference Experiments. Political Analysis 22(1):1-30

See Also

`amce`

Examples

```
data("immigrationconjoint")
## Not run:
## You can load a design from a .dat file from the Conjoint SDT
immigrationdesign <- makeDesign(type="file", filename="immigrant.dat")

## End(Not run)

## Or you can construct the conjoint design manually in in R
attribute_list <- list()
attribute_list[["Education"]] <-c("no formal","4th grade",
                                  "8th grade","high school",
                                  "two-year college","college degree",
                                  "graduate degree")
attribute_list[["Gender"]] <- c("female","male")
attribute_list[["Country of Origin"]] <-  c("Germany","France","Mexico",
                                            "Philippines","Poland","India",
                                            "China","Sudan","Somalia","Iraq")
attribute_list[["Reason for Application"]] <- c("reunite with family",
                                                "seek better job",
                                                "escape persecution")
attribute_list[["Job"]] <- c("janitor","waiter","child care provider",
                             "gardener","financial analyst",
                             "construction worker","teacher",
                             "computer programmer","nurse",
                             "research scientist","doctor")
attribute_list[["Job Experience"]] <- c("none","1-2 years",
                                        "3-5 years","5+ years")
attribute_list[["Job Plans"]] <- c("contract with employer",
                                   "interviews with employer", "will look for work",
                                   "no plans to look for work")
attribute_list[["Prior Entry"]] <- c("never","once as tourist",
                                     "many times as tourist","six months with family",
                                     "once w/o authorization")
attribute_list[["Language Skills"]] <- c("fluent English",
                                         "broken English",
                                         "tried English but unable",
                                         "used interpreter")

# Randomization constraints in the conjoint design
constraint_list <- list()
# Constraints on Education and Job attributes
# Cannot have "doctor" with "no formal" - "high school" education
constraint_list[[1]] <- list()
constraint_list[[1]][["Education"]] <- c("no formal","4th grade",
                                         "8th grade","high school")
constraint_list[[1]][["Job"]] <- c("financial analyst",
                                   "computer programmer","research scientist",
                                   "doctor")
# Constraints on Reason for Application/Country of Origin
constraint_list[[2]] <- list()
constraint_list[[2]][["Reason for Application"]] <- c("escape persecution")
```

```
constraint_list[[2]][["Country of Origin"]] <- c("Germany","France",
                                                 "Mexico","Philippines",
                                                 "Poland","India")

immigrationdesign <- makeDesign(type='constraints', attribute.levels=attribute_list,
                                constraints=constraint_list)

## You can set your own user-defined distribution over the attribute-levels
marginal_weights <- list() # Uniform for all except education which is bimodal
marginal_weights[["Education"]] <- c(1/20, 1/20, 2/10, 2/10, 1/10, 2/10, 2/10)
marginal_weights[["Gender"]] <- rep(1/length(attribute_list[["Gender"]]),
                                length(attribute_list[["Gender"]]))
marginal_weights[["Country of Origin"]] <- rep(1/length(
                                      attribute_list[["Country of Origin"]]),
                                      length(attribute_list[["Country of Origin"]]))
marginal_weights[["Reason for Application"]] <- rep(1/length(
                               attribute_list[["Reason for Application"]]),
                               length(attribute_list
                               [["Reason for Application"]]))
marginal_weights[["Job"]] <- rep(1/length(attribute_list[["Job"]]),
                               length(attribute_list[["Job"]]))
marginal_weights[["Job Experience"]] <- rep(1/length(attribute_list[["Job Experience"]]),
                                      length(attribute_list[["Job Experience"]]))
marginal_weights[["Job Plans"]] <- rep(1/length(attribute_list[["Job Plans"]]),
                                      length(attribute_list[["Job Plans"]]))
marginal_weights[["Prior Entry"]] <- rep(1/length(attribute_list[["Prior Entry"]]),
                                      length(attribute_list[["Prior Entry"]]))
marginal_weights[["Language Skills"]] <- rep(1/length(attribute_list[["Language Skills"]]),
                                      length(attribute_list[["Language Skills"]]))

## Not run:
immigrationdesign_reweight <- makeDesign(type='constraints', attribute.levels=attribute_list,
                               constraints=constraint_list, level.probs=marginal_weights)

## Note that estimated AMCEs can depend on the randomization distribution

results <- amce(Chosen_Immigrant ~  Gender + Education + Job, data=immigrationconjoint,
                cluster=TRUE, respondent.id="CaseID", design=immigrationdesign)
summary(results)

results_wt <- amce(Chosen_Immigrant ~  Gender + Education + Job, data=immigrationconjoint,
                cluster=TRUE, respondent.id="CaseID", design=immigrationdesign_reweight)
summary(results_wt)

## End(Not run)
```

`plot.amce` *Plot AMCE Estimates*

Description

`plot` method for "amce" objects

Usage

```
## S3 method for class 'amce'
plot(x, main = "", xlab = "Change in E[Y]", ci = 0.95,
                    colors = NULL, xlim = NULL, breaks = NULL,
                    labels = NULL, attribute.names = NULL, level.names = NULL,
                   label.baseline = TRUE, text.size = 11, text.color = "black",
                    point.size = 0.5, dodge.size = 0.9, plot.theme = NULL,
                    plot.display = "all",
facet.names = NULL, facet.levels = NULL,
group.order = NULL,font.family = NULL, ...)
```

Arguments

x	An object of class "amce", a result of a call to amce
main	Title of the plot.
xlab	Label of the x-axis of the plot (AMCE or ACIE). Default is "Change in E[Y]"
ci	Levels for confidence intervals to plot around point estimates. Must be between 0 and 1. Default is .95
colors	Vector of color names to be used for points and confidence intervals. The plot function will alternate between the colors in the vector for each attribute being plotted. If NULL, plot will use a default ggplot2 color scheme.
xlim	Numeric vector denoting the upper and lower bounds of the x-axis in the plot. If NULL the plot function will automatically set a range that includes all effect estimates.
breaks	Numeric vector denoting where x-axis tick marks should be placed. If NULL plot will use ggplot2 defaults.
labels	Vector denoting how x-axis tick marks should be labeled. If NULL plot will use ggplot2 defaults.
attribute.names	Character vector of attribute names to be plotted as labels. By default plot.amce will use the attribute names in the "amce" object passed to it.
level.names	A list containing character vector elements with names in attribute.names. Each character vector in the list contains the level names to be plotted as labels beneath the corresponding attribute. By default plot.amce will use the level names in the "amce" object passed to it.
label.baseline	If TRUE, the baseline levels for each attribute will be labeled as such. Defaults to TRUE.
text.size	Size of text. Defaults to 11.
text.color	Color of text in plot. Defaults to "black".
point.size	Size of points in the plot. Defaults to 0.5.
dodge.size	Width to dodge overlaps to the side. Defaults to 0.9.
plot.theme	A ggplot2 'theme' object to be added to the plot. If NULL, defaults to black-and-white theme. Note that passing a theme object will override text and point color/size options.

plot.display	Character string, one of "all", "unconditional", or "interaction". Option "all" will display both unconditional and interaction estimates. The "unconditional" option will display only 1 plot for unconditional estimates (both AMCE and ACIE) ignoring any facets provided to "facet.names" or respondent-varying characteristics. Option "interaction" will drop the unconditional plot and instead display only (unconditional) ACIE's or estimates conditional on respondent-varying characteristics as specified in the user-supplied option "facet.names". Defaults to "all".
facet.names	To facet plots (i.e., make separate plots for each value of a variable) give "facet.names" a vector of character strings containing the names of the variable(s) (either profile attribute or respondent-varying) to facet by. Unless given specific levels in "facet.levels", the plotted levels will consist of all levels of a factor variable or the quantiles of a continuous variable. Multiple facet variables cannot currently be varied at the same time within the same plot. Instead conditional effects will be calculated for one facet at a time while others are held at their first value in facet.levels, by default the bottom quantile for continuous variables and the baseline for factors.
facet.levels	To manually set facet levels, provide a list to "facet.levels". Names of list entries should correspond with variable names. The content of each entry should be a vector giving the desired levels, whether factors or continuous. To change the displayed names of the levels, assign names to each vector entry.
group.order	To manually set the order of the attributes, provide a vector to "group.order". Names of the vector entries should correspond with name of the attribute.
...	Other graphical parameters passed to ggplot.
font.family	Will be passed to the ggplot function as the argument for font.family. If NULL, defaults will be used.

Value

A ggplot object containing a dotplot of estimated AMCEs.

References

Hainmueller, J., Hopkins, D., and Yamamoto T. (2014) Causal Inference in Conjoint Analysis: Understanding Multi-Dimensional Choices via Stated Preference Experiments. Political Analysis 22(1):1-30

See Also

amce for the main estimation routine.

Examples

```
## Not run:
# Immigration Choice Conjoint Experiment Data from Hainmueller et. al. (2014).
data("immigrationconjoint")
data("immigrationdesign")
```

```
# Run AMCE estimator using all attributes in the design
results <- amce(Chosen_Immigrant ~  Gender + Education + `Language Skills` +
                `Country of Origin` + Job + `Job Experience` + `Job Plans` +
                `Reason for Application` + `Prior Entry`, data=immigrationconjoint,
                cluster=TRUE, respondent.id="CaseID", design=immigrationdesign)

# Plot results
plot(results, xlab="Change in Pr(Immigrant Preferred for Admission to U.S.)",
    ylim=c(-.3,.3), breaks=c(-.2, 0, .2), labels=c("-.2","0",".2"), text.size=13)

# Plot results with user-specified order of attributes
plot(results, xlab="Change in Pr(Immigrant Preferred for Admission to U.S.)",
     ylim=c(-.3,.3), breaks=c(-.2, 0, .2), labels=c("-.2","0",".2"), text.size=13,
     group.order=c("Gender","Education","Job",
                     "Language Skills","Job Experience",
                     "Job Plans","Reason for Application",
                     "Prior Entry","Country of Origin"))


# Run AMCE estimator with an interaction with a respondent-varying characteristic
interaction_results <- amce(Chosen_Immigrant ~ Gender + Education
    + Job + ethnocentrism:Job,
    data = immigrationconjoint,na.ignore=TRUE,
    design = immigrationdesign, cluster = FALSE,
    respondent.varying = "ethnocentrism")

# Plot results with additional plots for quantiles of the respondent-varying characteristic
plot(interaction_results)

# Plot results with user-specified order of attributes
plot(interaction_results, group.order=c("Gender","Education","Job"))

# Do not show output for variables that do not vary with facetted levels
plot(interaction_results,plot.display="unconditional")

# RUN AMCE estimator with an interaction between two factor variables
interaction_results <- amce(Chosen_Immigrant ~ Gender + Education + Job
    + Education:Job, data = immigrationconjoint,
    cluster = FALSE, design = immigrationdesign)

# Plot results with different plots for all levels of ACIE
plot(interaction_results,facet.names = "Education")


# Plot results with different plots for only two levels of one interacted variable
facet.levels1 <- list()
facet.levels1[["Education"]] <- c("college degree","graduate degree")
plot(interaction_results,facet.names = "Education",facet.levels = facet.levels1)

# Display only interaction panes
plot(interaction_results,facet.names = "Education",plot.display="interaction")

#Display only non-interaction panes
```

```
plot(interaction_results,facet.names = "Education",plot.display="unconditional")

#Change displayed attribute and level names
results <- amce(Chosen_Immigrant ~ Gender + Education + Job, data = immigrationconjoint,
    cluster = FALSE, design = immigrationdesign)
levels.test<-list()
levels.test[["Gender"]]<-c("level1","level2")
levels.test[["Education"]]<-c("level1","b","c","d","e","f","g")
levels.test[["Job"]]<-c("a","b","c","d","e","f","g","h","i","j","k")
plot(results, level.names = levels.test, main="test", xlab="test",
ci=0.95, breaks=c(-0.2,-0.1,0,0.1,0.2), attribute.names = c("attribute1","attribute2","attribute3"))


## End(Not run)
```

`read.qualtrics` *Read Data from a Conjoint Qualtrics Experiment*

Description

Converts the raw .CSV data file downloaded from an online conjoint experiment run using the Qualtrics survey software into a data frame usable by the `amce` routine. Each row of the Qualtrics .CSV file is a single survey respondent. The rows of the resulting dataframe correspond to individual profile choices. Currently, the function supports various types of outcomes in a conjoint experiment, with details explained below.

Usage

```
read.qualtrics(filename, responses = NULL, covariates = NULL,
respondentID = NULL, letter = "F",  new.format=FALSE, ranks = NULL)
```

Arguments

`filename`	A .CSV file containing responses exported from a Qualtrics survey experiment. The first row contains question and variable identifiers (e.g. V1, Q1). The second row contains variable names and question texts. Subsequent rows contain the answers of each respondent. This .CSV file can be exported through the "Download Data" panel in the Qualtrics web interface. Note that answers should be shown as "coded values" and not as choice text.
`responses`	A character or integer vector with the identifiers of the CSV columns that contain the conjoint responses. The first element corresponds to the identifier of the first question, the second element corresponds to the second question and so on. The length of this vector must be equal to the number of conjoint tasks answered by each respondent. Identifiers are typically in the form "Q#" - e.g. `c("Q1", "Q3", "Q5")` would represent a three question survey where the conjoint questions are Q1, Q3 and Q5. If specified as an integer vector, the elements are the column numbers corresponding to each question response.

`covariates`	A character vector denoting the column names of any other respondent-level characteristics measured in the survey that should be included in the resulting dataframe.
`respondentID`	A character string denoting a column containing a unique identifier for each respondent (e.g. an IP address). This identifier will be carried over into the output. If NULL, each respondent will be given an arbitrary identifier in the output dataframe. Leave as NULL if you do not want responses to be linked back to a known respondent identifier.
`new.format`	Indicator for whether the .csv file is from the new Qualtrics export format with three title rows (TRUE) or from the old format (FALSE) with two title rows. Defaults to FALSE.
`ranks`	An integer vector with the identifiers of the CSV columns that contain the conjoint rankings or ratings.
`letter`	The beginning letter used in the naming convention of levels and attributes.

Details

This function currently only works with experiments that generate profiles using .PHP scripts created by the Conjoint Survey Design Tool. It also is only able to handle standard conjoint designs (binary outcome variable/forced choice).
(https://github.com/astrezhnev/conjointsdt/).

For each respondent in the .CSV file, attribute and level names are stored using the following naming convention:
Level Name: F-[task number]-[profile number]-[attribute number]
Attribute Name: F-[task number]-[attribute number]
Example: F-1-3-2 denotes the level corresponding to Task 1, Profile 3, Attribute 2
F-3-3 denotes the attribute name corresponding to Task 3, Attribute 3

Special Characters: Some special characters have been reserved for this function if (cjoint>v2.0.6) for efficienty purpose. This means some special characters cannot be used in the attribute names in your data. However, if you still want to display special characters in attribute names in your plot, you may want to use the argument `attribute.names` in function `plot.amce` to customize the display of your attribute names.

East Asian Language Support: The read.qualtrics function relies on the read.csv function in R-core. The read.csv function only works well for some of character encoding, but not others, for East Asian languages in some Operation systems. In Windows, .csv files containing East Asian languages such as Chinese or Japanese should be stored in the ANSI encoding rather than UTF-8 for the read.csv function to work. Further, if you are reading the csv file in a Windows OS with a different display language as the file you are trying to read in, you will need to reconfigure R into the language of the csv file by `Sys.setlocale()`.

Different types of responses: This function supports various types of responses commonly used in conjoint analyses. Here are some illustrations on some typical types.

1. The respondent is asked to fill in the profile she prefers the most, and her choice is restored in one response variable. In this case, set the argument responses=the response variable.

2. The respondent is asked to give each profile a rank within each task, and her ranks for each profile within each task are restored in J response variables, suppose there are J profiles within each

task. In this case, set the argument ranks as a vector restoring the variables names of these responses variables, in the order of rank of profile 1 in task 1, rank of profile 2 in task 1, ... rank of profile J in task 1, rank of profile 1 in task 2, rank of profile 2 in task 2, ... rank of profile J in task 2, ..., rank of profile J in task K.

3. The respondent is asked to rate each profile within each task, and her ratings for each profile within each task are restored in J response variables, suppose there are J profiles within each task. In this case, set the argument ratings as a vector restoring the variables names of these responses variables, in the order of rating of profile 1 in task 1, rating of profile 2 in task 1, ... rating of profile J in task 1, rating of profile 1 in task 2, rating of profile 2 in task 2, ... rating of profile J in task 2, ..., rating of profile J in task K.

4. The respondent is asked to select the top L profiles she prefers within each task. L colud be smaller than the number of profiles J available within each task. Her choices are recorded with L variables for each task, indicating the first choice among J profiles in task 1, the second choice among J profiles in task 1, ..., the L-th choice among J profiles in task 1, the first choice among J profiles in task 2, the second choice among J profiles in task 2, ..., the L-th choice among J profiles in task 2, ..., the L-th choice among J profiles in task K. In this case, the read.qualtrics function should be applied in the following step:

(a) Set the argument response as a vector restoring the variables names of the first choice among J profiles in all tasks, in the order of first choice among J profiles in task 1, first choice among J profiles in task 2, ..., first choice among J profiles in task K. Save the respective data output.

(b) Set the argument response as a vector restoring the variables names of the second choice among J profiles in all tasks, in the order of second choice among J profiles in task 1, second choice among J profiles in task 2, ..., second choice among J profiles in task K. Save the respective data output.

(c) Repeat until you have completed the above steps for all top L choices. And then merge all data according to their respondentID. However, you may need to change the variable name of the responses a little bit for a successful merge.

5. The respondent is asked to select the top L profiles she prefers within each task. L colud be smaller than the number of profiles J available within each task. The questionnaire is designed in such format that the respondent fills in the ranking of top L profiles that she prefers, while leave the ranking of the other J-L profiles blank. In this case, set the argument ranks as a vector restoring the variables names of these J responses variables, in the order of rank of profile 1 in task 1, rank of profile 2 in task 1, ... rank of profile J in task 1, rank of profile 1 in task 2, rank of profile 2 in task 2, ... rank of profile J in task 2, ..., rank of profile J in task K.

Value

A dataframe in which each row corresponds to a single profile. The column "selected" denotes whether that profile was selected by the respondent. The columns "respondent" and "task" denote the respondent and task numbers to which the profile was assigned. Respondent-level covariates are appended to each row.

References

Strezhnev, A., Hainmueller, J., Hopkins, D., and Yamamoto, T. (2014) Conjoint Survey Design Tool. https://github.com/astrezhnev/conjointsdt/

Examples

```
## Not run:
# An example file with 5 conjoint tasks per respondent and a single covariate
# You can demonstrate this function's output using the CandidateConjointQualtrics.csv
# demonstration file in the 'inst' subdirectory
conjoint_data <- read.qualtrics("CandidateConjointQualtrics.csv",
responses=c("Q2.3", "Q2.7", "Q2.10", "Q2.13", "Q2.16"),
covariates=c("Q6.6"), respondentID="V1")

## End(Not run)
```

`read.with.qualtRics` *Read Data from a Conjoint Qualtrics Experiment*

Description

Converts the raw dataframe downloaded with R Package `"qualtRics"` from an online conjoint experiment into a data frame usable by the `amce` routine. Each row of the raw dataframe is a single survey respondent. The rows of the resulting dataframe correspond to individual profile choices. Currently, the function supports various types of outcomes in a conjoint experiment, with details explained below.

Usage

```
read.with.qualtRics(filename, responses = NULL, covariates = NULL,
respondentID = NULL, letter = "F",  new.format=FALSE, ranks = NULL)
```

Arguments

`filename`	A dataframe containing responses download from a Qualtrics survey experiment with R package `"qualtRics"`. Each row contains the answers of each respondent. Note that answers should be shown as "coded values" and not as choice text.
`responses`	A character or integer vector with the identifiers of the CSV columns that contain the conjoint responses. The first element corresponds to the identifier of the first question, the second element corresponds to the second question and so on. The length of this vector must be equal to the number of conjoint tasks answered by each respondent. Identifiers are typically in the form "Q#" - e.g. `c("Q1", "Q3", "Q5")` would represent a three question survey where the conjoint questions are Q1, Q3 and Q5. If specified as an integer vector, the elements are the column numbers corresponding to each question response.
`covariates`	A character vector denoting the column names of any other respondent-level characteristics measured in the survey that should be included in the resulting dataframe.

respondentID	A character string denoting a column containing a unique identifier for each respondent (e.g. an IP address). This identifier will be carried over into the output. If NULL, each respondent will be given an arbitrary identifier in the output dataframe. Leave as NULL if you do not want responses to be linked back to a known respondent identifier.
new.format	Indicator for whether the .csv file is from the new Qualtrics export format with three title rows (TRUE) or from the old format (FALSE) with two title rows. Defaults to FALSE.
ranks	An integer vector with the identifiers of the CSV columns that contain the conjoint rankings or ratings.
letter	The beginning letter used in the naming convention of levels and attributes.

Details

This function currently only works with experiments that generate profiles using .PHP scripts created by the Conjoint Survey Design Tool. It also is only able to handle standard conjoint designs (binary outcome variable/forced choice).
(https://github.com/astrezhnev/conjointsdt/).

For each respondent in the dataframe, attribute and level names are stored using the following naming convention:
Level Name: F-[task number]-[profile number]-[attribute number]
Attribute Name: F-[task number]-[attribute number]
Example: F-1-3-2 denotes the level corresponding to Task 1, Profile 3, Attribute 2
F-3-3 denotes the attribute name corresponding to Task 3, Attribute 3

East Asian Language Support: The read.qualtrics function relies on the read.csv function in R-core. The read.csv function only works well for some of character encoding, but not others, for East Asian languages in some Operation systems. In Windows, .csv files containing East Asian languages such as Chinese or Japanese should be stored in the ANSI encoding rather than UTF-8 for the read.csv function to work.

Different types of responses: This function supports various types of responses commonly used in conjoint analyses. Here are some illustrations on some typical types.

1. The respondent is asked to fill in the profile she prefers the most, and her choice is restored in one response variable. In this case, set the argument responses=the response variable.

2. The respondent is asked to give each profile a rank within each task, and her ranks for each profile within each task are restored in J response variables, suppose there are J profiles within each task. In this case, set the argument ranks as a vector restoring the variables names of these responses variables, in the order of rank of profile 1 in task 1, rank of profile 2 in task 1, ... rank of profile J in task 1, rank of profile 1 in task 2, rank of profile 2 in task 2, ... rank of profile J in task 2, ..., rank of profile J in task K.

3. The respondent is asked to rate each profile within each task, and her ratings for each profile within each task are restored in J response variables, suppose there are J profiles within each task. In this case, set the argument ratings as a vector restoring the variables names of these responses variables, in the order of rating of profile 1 in task 1, rating of profile 2 in task 1, ... rating of profile J in task 1, rating of profile 1 in task 2, rating of profile 2 in task 2, ... rating of profile J in task 2, ..., rating of profile J in task K.

4. The respondent is asked to select the top L profiles she prefers within each task. L colud be smaller than the number of profiles J available within each task. Her choices are recorded with L variables for each task, indicating the first choice among J profiles in task 1, the second choice among J profiles in task 1, ..., the L-th choice among J profiles in task 1, the first choice among J profiles in task 2, the second choice among J profiles in task 2, ..., the L-th choice among J profiles in task 2, ..., the L-th choice among J profiles in task K. In this case, the read.qualtrics function should be applied in the following step:

(a) Set the argument response as a vector restoring the variables names of the first choice among J profiles in all tasks, in the order of first choice among J profiles in task 1, first choice among J profiles in task 2, ..., first choice among J profiles in task K. Save the respective data output.

(b) Set the argument response as a vector restoring the variables names of the second choice among J profiles in all tasks, in the order of second choice among J profiles in task 1, second choice among J profiles in task 2, ..., second choice among J profiles in task K. Save the respective data output.

(c) Repeat until you have completed the above steps for all top L choices. And then merge all data according to their respondentID. However, you may need to change the variable name of the responses a little bit for a successful merge.

5. The respondent is asked to select the top L profiles she prefers within each task. L colud be smaller than the number of profiles J available within each task. The questionnaire is designed in such format that the respondent fills in the ranking of top L profiles that she prefers, while leave the ranking of the other J-L profiles blank. In this case, set the argument ranks as a vector restoring the variables names of these J responses variables, in the order of rank of profile 1 in task 1, rank of profile 2 in task 1, ... rank of profile J in task 1, rank of profile 1 in task 2, rank of profile 2 in task 2, ... rank of profile J in task 2, ..., rank of profile J in task K.

Value

A dataframe in which each row corresponds to a single profile. The column "selected" denotes whether that profile was selected by the respondent. The columns "respondent" and "task" denote the respondent and task numbers to which the profile was assigned. Respondent-level covariates are appended to each row.

References

Strezhnev, A., Hainmueller, J., Hopkins, D., and Yamamoto, T. (2014) Conjoint Survey Design Tool. https://github.com/astrezhnev/conjointsdt/

Examples

```
## Not run:
# An example file with 5 conjoint tasks per respondent and a single covariate
# Suppose "CandidateConjointQualtrics" is the dataframe exported by R Package qualtRics
conjoint_data <- read.with.qualtRics(CandidateConjointQualtrics,
responses=c("Q2.3", "Q2.7", "Q2.10", "Q2.13", "Q2.16"),
covariates=c("Q6.6"), respondentID="V1")

## End(Not run)
```

summary.amce *Summarizing AMCE estimates*

Description

summary method for class "amce"

Usage

```
## S3 method for class 'amce'
summary(object, covariate.values=NULL, ...)

## S3 method for class 'summary.amce'
print(x, digits=5, ...)
```

Arguments

object	An object of class "amce", a result of a call to amce.
covariate.values	An optional list containing a vector at which conditional effects will be calculated in the case of AMCE and ACIE's conditional on respondent-varying characteristics. The class of the values in the vector must match the class of the respondent-varying characteristic in question. If the "amce" object contains respondent varying characteristics, when set to NULL (default) interaction effects will be reported at quantiles in the case of a continuous variable and levels in the case of a factor. Names of list entries must correspond to variable names. If there are multiple respondent-varying characteristics then while each is varied in turn, all others will be held at first value of their entry in covariate.values. This is the bottom quantile in the case of a continuous variable and the baseline in the case of a factor variable.
x	An object of class "summary.amce", a result of a call to summary.amce.
digits	The number of significant digits to use when printing.
...	Further arguments from other methods.

Value

The function summary.amce computes and returns formatted data frames of effect estimates returned by amce

amce	A dataframe containing AMCE estimates and standard errors. Each row corresponds to a single attribute-level effect.
baselines_amce	Baseline levels for each attribute relative to which the AMCEs are calculated.
acie	A dataframe containing ACIE estimates and standard errors, if any. Each row corresponds to a single attribute-level effect.
baselines_acie	Baseline levels for each attribute relative to which the ACIEs are calculated.

`baselines_amce_resp`
: Baseline levels for conditional AMCE estimates, if any, relative to which interactions are calculated.

`baselines_acie_resp`
: Baseline levels for conditional ACIE estimates, if any, relative to which interactions are calculated.

`samplesize_estimates`
: The number of valid profiles (rows) in the dataset when only effects of profile varying attributes are calculated.

`samplesize_resp`
: The number of valid profiles (rows) in the dataset when respondent-varying characteristics are incorporated.

`numrespondents`
: The number of respondents in the dataset (if a `respondent.id` argument was passed to `amce`)

`table_values_amce`
: A dataframe giving the names of additional tables of AMCE estimates conditional on respondent-varying characteristics. A separate table is produced for each level of each respondent varying characteristic.

`table_values_acie`
: A dataframe giving the names of tables of ACIE estimates conditional on respondent-varying characteristics. A separate table is produced for each level of each respondent varying characteristic.

See Also

The estimation function `amce`.

Examples

```
## Not run:

#Results with respondent-varying characteristics
results <-amce(Chosen_Immigrant ~ Gender + Education + Education:ethnocentrism +
`Country of Origin`+ `Country of Origin`:ethnocentrism + Job +
Job:ethnocentrism + `Job Experience` + `Job Experience`:ethnocentrism,
data=immigrationconjoint, design=immigrationdesign,cluster=FALSE,
respondent.varying="ethnocentrism", na.ignore=TRUE)

#Calculate conditional estimates at user-supplied levels
int.vals<-list()
int.vals[["ethnocentrism"]]<-c(60,77,88,99,45)
summary(results, covariate.values = int.vals)


## End(Not run)
```

`view` *Graphical interfaces for conjoint analysis*

Description

Launches a graphical interface (in R Shiny) for a given conjoint analysis operation. This allows the user to interactively perform a conjoint analysis task without having to specify any code in R. Graphical interface is launched locally in user's default web browser. It is recommended that this feature be used for exploratory analysis only.

Examples

```
## Not run:
# Launch graphical interface to interactively run AMCE
view(amce)

## End(Not run)
```

Index