Package 'unvs.med'

August 23, 2026

Title A Universal Approach for Causal Mediation Analysis

Version 1.1.2

Date 2026-08-23

Description This program realizes a universal estimation approach that accommodates multi-category variables and effect scales, making up for the deficiencies of the existing approaches when dealing with non-binary exposures and complex models. The estimation via bootstrapping can simultaneously provide results of causal mediation on risk difference (RD), odds ratio (OR) and risk ratio (RR) scales with tests of the effects' difference.
The estimation is also applicable to many other settings, e.g., moderated mediation, inconsistent covariates,
panel data, etc. The high flexibility and compatibility
make it possible to apply for any type of model, greatly meeting the needs of current empirical researches.

Depends R (>= 3.5.0), data.table, parallel, snowfall

Suggests fixest, lme4, MASS, ordinal, pglm,

License GPL-3

Encoding UTF-8

LazyLoad yes

LazyData true

NeedsCompilation no

Repository CRAN

Config/build/clean-install true

Config/check/use_internal_timestamps true

Config/roxygen2/version 8.0.0

Author Tianbao Zhou [aut, cre],
Xinghao Li [aut]

Maintainer Tianbao Zhou <michaelzhou@buaa.edu.cn>

Date/Publication 2026-08-23 06:30:02 UTC

Contents

BootEstimation_for . . . 2
BootEstimation_MT . . . 3
cond_cov . . . 4
confirmingX . . . 5
FormalEstmed . . . 5
ident_M_type . . . 9
ident_Y_type . . . 10
plot.unvs.med . . . 11
potentialoutcome_facX . . . 12
potentialoutcome_numX . . . 13
SingleEstimation . . . 14
Statistics . . . 15
summary.unvs.med . . . 16
testdata . . . 17
um.test1 . . . 18
um.test2 . . . 19

Index **20**

`BootEstimation_for` *Bootstrapping Estimation for Causal Mediation Effects via Ordinary "for" Loop*

Description

This function obtains the estimates of mediation effects by the ordinary `for` loop. Through bootstrap sampling and repeating the algorithm of function `SingleEstimation`, This function obtains a number of estimates for each type of effect.

This is an internal function, automatically called by the function `Statistics`.

Usage

```
BootEstimation_for (m_model, y_model, data, X, exp0=NULL, exp1=NULL,
M, Y, m_type, y_type, boot_num = 100)
```

Arguments

`m_model`	a fitted model object for the mediator.
`y_model`	a fitted model object for the outcome.
`data`	a dataframe used in the analysis.
`X`	a character variable of the exposure's name.
`exp0`	a numeric variable of the baseline level of the exposure.
`exp1`	a numeric variable of the new level of the exposure.
`M`	a character variable of the mediator's name.

Y	a character variable of the outcome's name.
m_type	a character variable of the mediator's type.
y_type	a character variable of the outcome's type.
boot_num	the times of bootstrapping in the analysis. The default is 100.

Details

This function is realized by the ordinary `for` loop, therefore may take longer time to proceed. For small amounts of data, e.g., dozens to a hundred samples, with relatively simple models, `for` loop is recommended.

Value

This function returns a list of three dataframes, i.e., the bootstrapping results of the mediation effects. This list is also saved in the return of the main function `FormalEstmed`.

`BootEstimation_MT` *Bootstrapping Estimation for Causal Mediation Effects via Multi-threading Process*

Description

This function obtains the estimates of mediation effects via non-parametric bootstrapping. Through bootstrap sampling and repeating the algorithm of function `SingleEstimation`, This function obtains a number of estimates for each type of effect.

This is an internal function, automatically called by the function `Statistics`.

Usage

```
BootEstimation_MT (m_model, y_model, data, X, exp0=NULL, exp1=NULL,
M, Y, m_type, y_type, boot_num = 100)
```

Arguments

m_model	a fitted model object for the mediator.
y_model	a fitted model object for the outcome.
data	a dataframe used in the analysis.
X	a character variable of the exposure's name.
exp0	a numeric variable of the baseline level of the exposure.
exp1	a numeric variable of the new level of the exposure.
M	a character variable of the mediator's name.
Y	a character variable of the outcome's name.
m_type	a character variable of the mediator's type.
y_type	a character variable of the outcome's type.
boot_num	the times of bootstrapping in the analysis. The default is 100.

Details

This function activates the multi-threading process through package 'snowfall' in R with `max-1` cores (CPU) of the PC.

Value

This function returns a list of three dataframes, i.e., the bootstrapping results of the mediation effects. This list is also saved in the return of the main function `FormalEstmed`.

`cond_cov` *Modifications of Data and Models for Moderated Mediation*

Description

This function modifies the mediator and outcome model and the dataset according to the conditions of certain covariates specified by the user. The conditions are constructed by the character parameter `cov_val` with multiple string elements. This function then modifies the data and models based on rules of conditional regressions. The amount of sample and model structure are changed correspondingly. The samples and model variables satisfy the conditions are finally remain. Therefore, This function is only involved when moderated mediation effects are considered in the analysis.

This is an internal function, automatically called by the function `FormalEstmed`.

Usage

```
cond_cov (m_model, y_model, data, X, M, cov_val)
```

Arguments

`m_model`	a fitted model object for the mediator.
`y_model`	a fitted model object for the outcome.
`data`	a dataframe used in the analysis.
`X`	a character variable of the exposure's name.
`M`	a character variable of the mediator's name.
`cov_val`	a character variable of the conditions of the covariates. Each string (element) in the character may include ==, <, >, <=, >=, etc.

Value

This function returns a list of three objects. A conditional dataframe, an updated mediator model and an updated outcome model.

confirmingX *Confirmation of the Exposure*

Description

This function checks the specification of the exposure in function FormalEstmed specified by users. The name of exposure specified in function FormalEstmed should be a character. The exposure should be a numeric and binary factor, otherwise this function warns.

This is an internal function, automatically called by the function FormalEstmed.

Usage

```
confirmingX (X)
```

Arguments

X	a character variable of the exposure's name.

Value

No return value, called for checking the exposure's name.

FormalEstmed *Formal Estimation for Causal Mediation Effects (The Main Function)*

Description

This is the main function for causal mediation estimations. Users only need to call this function to estimate causal mediation effects, as it can automatically call other internal functions for the algorithm. This function provides estimates of various types of mediation effects on risk difference (RD), odds ratio (OR) and risk ratio (RR) scales in the returned object, in which a wide range of estimation details and model information are also included. This function is applicable to almost any type of mediator and outcome models and data structure, greatly increasing the efficiency of causal mediation analysis.

Usage

```
FormalEstmed (med_model, out_model, data, exposure, exp0=NULL, exp1=NULL,
mediator=NULL, outcome=NULL, med_type=NULL,out_type=NULL, cov_val=NULL,
boot_num=100, MT = TRUE, Cf_lv=0.95)
```

Arguments

med_model	a fitted model object for the mediator.
out_model	a fitted model object for the outcome.
data	a dataframe used in the analysis.
exposure	a character variable of the exposure's name. Must be specified by the user.
exp0	a numeric variable of the baseline level of the exposure.
exp1	a numeric variable of the new level of the exposure.
mediator	a character variable of the mediator's name. Identified automatically if not specified by the user.
outcome	a character variable of the outcome's name. Identified automatically if not specified by the user.
med_type	a character variable of the mediator's type. Identified automatically if not specified by the user.
out_type	a character variable of the outcome's type. Identified automatically if not specified by the user.
cov_val	a character variable of the conditions of the covariates. Each string (element) in the character variable is a logical statement, e.g., 'C1==5', 'C2>1', etc.
boot_num	the times of bootstrapping in the analysis. The default is 100.
MT	a logical value indicating whether the multi-threading process is activated. If TURE, activating max-1 cores. If FALSE, use the ordinary 'for' loop. The default is TRUE.
Cf_lv	a numeric variable of the confidence interval. The value is presented in decimal form, not percentage form. The default is 0.95.

Details

For continuous variables, mediator and outcome can be identified automatically by this function. However, when the mediator or outcome variable is not continuous, users should make sure the class of the variable is consistent both in the dataframe and in the models, otherwise, users should specify it manually, not automatically. For example, for a ordinal type of outcome variable, if users transfer it into a factor variable in the dataframe before building the model, outcome can be identified automatically. If users do not transfer it into a factor variable in advance, but only specify it as a factor within the model, e.g., polr(as.factor(outcome)~X1+X2+...), then outcome can not be identified automatically. Therefore we recommend users transfer the mediator and outcome variable properly in the dataframe before building models.

Value

This function returns a list object of class "unvs.med". The object encompasses the complete result of the estimates of various types of effects on risk difference (RD), odds ratio (OR) and risk ratio (RR) scales, the results of mom-parametric bootstrapping, model specifications and other detailed information. Users may conduct further analysis based on this object.

The function summary.unvs.med can be used to obtain the refined result of this returned object, The function plot.unvs.med can be used to obtain the visualized result of this returned object.

The function `um.test1` can be used to test the statistical difference of effects within one single estimation. The function `um.test2` can be used to test the statistical difference of effects between two separate estimations.

`Stat.RD`, `Stat.OR`, `Stat.RR`
: Statistics of the estimates of mediation effects on risk difference (RD), odds ratio (OR) and risk ratio (RR) scales.

`Boot_result`
: results of the original non-parametric bootstrapping estimations of mediation effects risk difference (RD), odds ratio (OR) and risk ratio (RR) scales.

`Function_call`
: user's code of calling function FormalEstmed().

`Exposure`
: exposure in the analysis.

`Mediator`
: mediator in the analysis.

`Medaitor_type`
: mediator's type in the analysis.

`Medaitor_model`
: mediator's model in the analysis. If involving moderated mediation, i.e., `Covariates_cond` is not `NULL`, then this returned mediator's model is different from the input one.

`Outcome`
: outcome in the analysis.

`Outcome_type`
: outcome's type in the analysis.

`Outcome_model`
: outcome's model in the analysis. If involving moderated mediation, i.e., `Covariates_cond` is not `NULL`, then this returned outcome's model is different from the input one.

`Covariates`
: covariates in the analysis.

`Covariates_cond`
: conditions of the covariates in the analysis.

`Data`
: dataframe used in the analysis. If involving moderated mediation, i.e., `Covariates_cond` is not `NULL`, then this returned dataframe is different from the input one.

`Bootstrap_number`
: times of bootstrapping in the analysis.

`Confidence_level`
: levels of confident interval in the analysis.

Note

The running time of this function depends on the quantity of data samples and the complexity of the mediator and outcome models. For example, the running time in the case of continuous mediator is significantly longer than that in the case of binary and ordinal mediator. For a certain type of mediator, it takes a longer time to proceed in the case of ordinal outcome. The running time in different settings also varies significantly, from a couples of seconds to several minutes. Therefore, we welcome users to provide a more efficient algorithm for the case of continuous mediators and contact our maintainer with no hesitation. We are looking forward to your suggestions and comments.

Examples

```
############################################################
# Example 1.1: Continuous exposure and outcome; Binary mediator
############################################################
data(testdata)
med_model=glm(med~exp+C1+C2+C3, data=testdata, family=binomial) # Fitting mediator's model
```

```
out_model=lm(out~med*exp+C1+C2+C3, data=testdata) # Fitting outcome's model
r11 = FormalEstmed (med_model=med_model, out_model=out_model,
data=testdata, exposure = "exp") # Running formal estimation via bootstrapping
summary(r11) # Viewing results in short form and on RD scales.

########
# the new level of the exposure at 3.5
# r11.1 and r11.2 are identical
r11.1 = FormalEstmed (med_model=med_model, out_model=out_model, data=testdata,
exposure = "exp", exp0 = 2.5)
summary(r11.1)
r11.2 = FormalEstmed (med_model=med_model, out_model=out_model, data=testdata,
exposure = "exp", exp0 = 2.5, exp1 = 3.5)
summary(r11.2)

# Setting the baseline level of the exposure at 5,
# the new level of the exposure at 7
r11.3 = FormalEstmed (med_model=med_model, out_model=out_model, data=testdata,
exposure = "exp", exp0 = 5, exp1 = 7)
summary(r11.3)

############################################################
# Example 1.2: Example 1.1 but considering moderated mediation
############################################################
data(testdata)
med_model=glm(med~exp*C1+C2+exp*C3, data=testdata, family=binomial) # Fitting mediator's model
out_model=lm(out~med*exp+exp*C1+C2+exp*C3, data=testdata) # Fitting outcome's model
r12 = FormalEstmed (med_model=med_model, out_model=out_model,
data=testdata, exposure = "exp", cov_val=c("C1==1","C3>7")) # Conditional on C1 and C3.
summary(r12)

############################################################
# Example 1.3: Example 1.1 with more bootstrapping
############################################################
data(testdata)
med_model=glm(med~exp+C1+C2+C3, data=testdata, family=binomial) # Fitting mediator's model
out_model=lm(out~med*exp+C1+C2+C3, data=testdata) # Fitting outcome's model
r13 = FormalEstmed (med_model=med_model, out_model=out_model,
data=testdata, exposure = "exp", boot=500) # Running formal estimation via bootstrapping
summary(r13) # Viewing results in short form and on RD scales.

############################################################
# Example 2.1: Continuous exposure; Binary mediator; Ordinal outcome
############################################################'
library("MASS") # For ordinal logistic regression
data(testdata)
med_model=glm(med~exp+C1+C2+C3, data=testdata, family=binomial) # Fitting mediator's model
testdata$out2=as.factor(testdata$out2) # out2 is the outcome. Convert it into a factor.
out_model=polr(out2~med*exp+C1+C2+C3, data=testdata, method="logistic") # Fitting outcome's model.
r21 = FormalEstmed (med_model=med_model, out_model=out_model,
data=testdata, exposure = "exp", boot=100) # Running formal estimation via bootstrapping.
summary(r21)
```

```
############################################################
# Example 2.2: Example 2.1 but considering moderated mediation
############################################################'
library("MASS") # For ordinal logistic regression
data(testdata)
med_model=glm(med~exp+C1+C2+C3, data=testdata, family=binomial) # Fitting mediator's model
testdata$out2=as.factor(testdata$out2) # out2 is the outcome. Convert it into a factor.
out_model=polr(out2~med*exp+C1+C2+C3, data=testdata, method="logistic") # Fitting outcome's model.
r22 = FormalEstmed (med_model=med_model, out_model=out_model,
data=testdata, exposure = "exp", boot=100, cov_val="C2>=50") # Solely conditioning on C2
summary(r22)

############################################################
# Example 3.1: Binary exposure (0 and 1); Binary mediator; continue outcome
############################################################'
data(testdata)
med_model=glm(med~exp2+C1+C2+C3, data=testdata, family=binomial) # Fitting mediator's model
out_model=lm(out~med*exp2+C1+C2+C3, data=testdata) # Fitting outcome's model
r31 = FormalEstmed (med_model=med_model, out_model=out_model,
data=testdata, exposure = "exp2") # Running formal estimation via bootstrapping
summary(r31) # Viewing results in short form and on RD scales.

############################################################
# Example 3.2: Binary exposure (male and female); Binary mediator; continue outcome
############################################################'
data(testdata)
med_model=glm(med~exp3+C1+C2+C3, data=testdata, family=binomial) # Fitting mediator's model
out_model=lm(out~med*exp3+C1+C2+C3, data=testdata) # Fitting outcome's model
r32 = FormalEstmed (med_model=med_model, out_model=out_model,
data=testdata, exposure = "exp3") # Running formal estimation via bootstrapping
summary(r32) # Viewing results in short form and on RD scales.

############################################################
# Example 3.3: Binary exposure (male and female); Binary mediator; continue outcome
############################################################'
data(testdata)
testdata$exp3=as.factor(testdata$exp3) # The default level is c("female","male").
levels(testdata$exp3)=c("male","female") # Factor level: c("male","female").
med_model=glm(med~exp3+C1+C2+C3, data=testdata, family=binomial) # Fitting mediator's model
out_model=lm(out~med*exp3+C1+C2+C3, data=testdata) # Fitting outcome's model
r33 = FormalEstmed (med_model=med_model, out_model=out_model,
data=testdata, exposure = "exp3") # Running formal estimation via bootstrapping
summary(r33) # Viewing results in short form and on RD scales.
```

---

ident_M_type *Identification of Mediator's type*

Description

This function identifies the mediator's type according to the model if the user does not directly specify it. The type of mediator could be `'continuous'`, `'binary'` or `'ordinal'`.

This is an internal function, automatically called by the function `FormalEstmed`.

Usage

```
ident_M_type (M, data)
```

Arguments

`M`	a character variable of the mediator's name.
`data`	a dataframe used in the analysis.

Value

This function returns a character as the mediator's type.

`ident_Y_type` *Identification of Outcome's type*

Description

This function identifies the outcome's type according to the model if the user does not directly specify it. The type of outcome could be `'ordinal'` or any other characters, e.g., 'non-ordinal'.

This is an internal function, automatically called by the function `FormalEstmed`.

Usage

```
ident_Y_type (Y, data)
```

Arguments

`Y`	a character variable of the outcome's name.
`data`	a dataframe used in the analysis.

Value

This function returns a character as the outcome's type.

plot.unvs.med *Visualization of Mediation Effects*

Description

This function is applied to the resulting object from function FormalEstmed to make the plot of the mediation effects. The plot is shown in boxplot.

Usage

```
## S3 method for class 'unvs.med'
 plot(x, scale = "RD", type = c("PNDE", "TNIE"), ...)
```

Arguments

x	a resulting object of class "unvs.med" from function FormalEstmed.
scale	a character variable of the effect scales. It may include more than one element. The default is "RD".
type	a character variable of the effect types. The character can contain any types. The default is c("PNDE", "TNIE").
...	additional parameters passed to "plot".

Details

(1) For instance, for parameter scale, users can specify it as "RD", c("RD"), c("RD", "OR"), c("RD", "OR", "RR"), etc. regardless of the order and quantity. Besides, if scale equals to "all", "ALL" or "All", then effects on three scales will be output (2) For type, the element is arbitrary, it may include any types of effects users are interested.

Value

No return value, called for making a plot.

Examples

```
# Running formal estimation
data(testdata)
med_model=glm(med~exp+C1+C2+C3, data=testdata, family=binomial) # Fitting mediator's model
out_model=lm(out~med*exp+C1+C2+C3, data=testdata) # Fitting outcome's model
r1 = FormalEstmed (med_model=med_model, out_model=out_model,
data=testdata, exposure = "exp") # Running formal estimation via bootstrapping

# Plot examples
plot(r1) # Plot of the default settings (plot PNED and TNDE on RD scales).
plot(r1,"OR") # Plot PNDE and TNIE on OR scales.
plot(r1,c("RD", "OR")) # Plot PNED and TNDE on RD and OR scales.
# Plot five types of the effects on RD and OR scales:
plot(r1,c("RD", "OR"), c("TE","PNDE", "TNDE", "PNIE", "TNIE"))
```

```
plot(r1,c("RD", "RR"), "all") # Plot all the effects on RD and RR scales.
plot(r1, scale=c("RD", "OR", "RR"), type="all") # Plot all the effects on all scales.
```

`potentialoutcome_facX` *Estimation of Potential Outcomes Based on the Universal Approach (for factor Exposure)*

Description

This function realizes the main algorithm of the universal approach to estimate potential outcomes with observed data. Different potential outcomes can be estimated by different combinations of the input parameters `xx` and `xm`.

This is an internal function, automatically called by the function `SingleEstimation`.

Usage

```
potentialoutcome_facX (xx, xm, data, X, M, Y,
m_type, y_type, m_model, y_model)
```

Arguments

`xx`	a counterfactual value for exposure, directly affecting the outcome. Equals 1 in the treatment group, equals 0 in the control group.
`xm`	a counterfactual value for exposure, directly affecting the mediator. Equals 1 in the treatment group, equals 0 in the control group.
`data`	a dataframe used for the above models in the mediation analysis.
`X`	a character variable of the exposure's name.
`M`	a character variable of the mediator's name.
`Y`	a character variable of the outcome's name.
`m_type`	a character variable of the mediator's type.
`y_type`	a character variable of the outcome's type.
`m_model`	a fitted model object for the mediator.
`y_model`	a fitted model object for the outcome.

Details

This function is called in the following cases. (1) The exposure is already specified as a factor variable in advance. (2) The exposure is a character variable. (3) The exposure is a character variable which the user already specify it as a factor variable with certain factor levels.

Value

This function returns a value of the potential outcome.

potentialoutcome_numX *Estimation of Potential Outcomes Based on the Universal Approach (for Numeric Exposure)*

Description

This function realizes the main algorithm of the universal approach to estimate potential outcomes with observed data. Different potential outcomes can be estimated by different combinations of the input parameters `xx` and `xm`.

This is an internal function, automatically called by the function `SingleEstimation`.

Usage

```
potentialoutcome_numX (xx, xm, data, X, exp0=NULL, exp1=NULL, M, Y,
m_type, y_type, m_model, y_model)
```

Arguments

`xx`	a counterfactual value for exposure, directly affecting the outcome. Equals 1 in the treatment group, equals 0 in the control group.
`xm`	a counterfactual value for exposure, directly affecting the mediator. Equals 1 in the treatment group, equals 0 in the control group.
`data`	a dataframe used for the above models in the mediation analysis.
`X`	a character variable of the exposure's name.
`exp0`	a numeric variable of the baseline level of the exposure.
`exp1`	a numeric variable of the new level of the exposure.
`M`	a character variable of the mediator's name.
`Y`	a character variable of the outcome's name.
`m_type`	a character variable of the mediator's type.
`y_type`	a character variable of the outcome's type.
`m_model`	a fitted model object for the mediator.
`y_model`	a fitted model object for the outcome.

Details

This function is called when the exposure is a numeric variable. Especially, a numeric exposure would be identified as a binary variable when it has only two values: 0 and 1. However, if the only two values are not 0 and 1, then users should specify it as a factor variable in advance so that function `potentialoutcome_facX` would be called automatically instead. The function still works well if the exposure is a multi-level discrete variable.

Value

This function returns a value of the potential outcome.

`SingleEstimation` *Single-time Estimation for Causal Mediation Effects*

Description

This function obtains the estimates of various types of mediation effects for a single time based on one bootstrap sample. This function calculates the effects through different combinations of potential outcomes by the algorithm we proposed.

This is an internal function, automatically called by the function `BootEstimation_for` or `BootEstimation_MT`.

Usage

```
SingleEstimation (m_model, y_model, data, X, exp0=NULL, exp1=NULL,
M, Y, m_type, y_type)
```

Arguments

`m_model`	a fitted model object for the mediator.
`y_model`	a fitted model object for the outcome.
`data`	a dataframe used in the analysis.
`X`	a character variable of the exposure's name.
`exp0`	a numeric variable of the baseline level of the exposure.
`exp1`	a numeric variable of the new level of the exposure.
`M`	a character variable of the mediator's name.
`Y`	a character variable of the outcome's name.
`m_type`	a character variable of the mediator's type.
`y_type`	a character variable of the outcome's type.

Value

This function returns a list of three dataframes of mediation effects on risk difference (RD), odds ratio (OR) and risk ratio (RR) scales. Each dataframe has only one single estimate for each effect.

`Statistics` *Statistics of Estimated Value of Causal Mediation Effects*

Description

This function calculates the statistics of mediation effects risk difference (RD), odds ratio (OR) and risk ratio (RR) scales based on the bootstrapping results. The statistics include the mean value, standard error, t-statistics, p-value and confident interval. The way to realize bootstrapping estimations is also specified in this function, either through the ordinal `for` loop, or the multi-threading process.

This is an internal function, automatically called by the function `FormalEstmed`.

Usage

```
Statistics (m_model, y_model, data, X, exp0=NULL, exp1=NULL, M, Y,
m_type, y_type, boot_num = 100, MT = TRUE, Cf_lv = 0.95)
```

Arguments

`m_model`	a fitted model object for the mediator.
`y_model`	a fitted model object for the outcome.
`data`	a dataframe used in the analysis.
`X`	a character variable of the exposure's name.
`exp0`	a numeric variable of the baseline level of the exposure.
`exp1`	a numeric variable of the new level of the exposure.
`M`	a character variable of the mediator's name.
`Y`	a character variable of the outcome's name.
`m_type`	a character variable of the mediator's type.
`y_type`	a character variable of the outcome's type.
`boot_num`	the times of bootstrapping in the analysis. The default is 100.
`MT`	a logical value indicating whether the multi-threading process is activated. If TURE, activating max-1 cores. If FALSE, use the ordinary 'for' loop. The default is `TRUE`.
`Cf_lv`	a numeric variable of the confidence interval. The value is presented in decimal form, not percentage form. The default is 0.95.

Details

This function also detects the treatment group and control group when the exposure is not a continuous variable. The function deals with o-1 exposure, character and factor exposure and then displays the grouping information for users.

Value

This function returns a list of three dataframes, i.e., statistics of the mediation effects risk difference (RD), odds ratio (OR) and risk ratio (RR) scales respectively. The statistics include the mean value, standard error, t-statistics, p-value and confident interval based on the bootstrapping estimations.

`summary.unvs.med` *Summary of Formal Estimation for Causal Mediation Effects*

Description

This function presents the refined statistics results of the mediation effects on risk difference (RD), odds ratio (OR) and risk difference (RR) scales from function `FormalEstmed`. This function is applied on the resulting object of class `"unvs.med"` from function `FormalEstmed`. The output shows the mean, standard error, t-statistics, p-value and confident interval of the effect estimates based on bootstrapping estimations.

Usage

```
## S3 method for class 'unvs.med'
 summary(object, form = "short", scale = "RD", ...)
```

Arguments

`object`	a resulting object of class `'unvs.med'` from function `FormalEstmed`.
`form`	a character variable indicating the output form. It can be `"short"` or `"long"`. The default is `"short"`.
`scale`	a character variable of the effect scales. It can be `"RD"`, `"OR"` or `"RR"`. If `scale` equals to string `"all"`, `"ALL"` or `"All"`, then effects on three scales will be displayed. The default is `"RD"`.
`...`	additional parameters passed to "summary". In addition, The function identifies it as the same string if the first letter is capital or all letters are capital. For example, `"short"` equals to `"Short"` and `"SHORT"`.

Value

No return value, called for displaying the output of the estimation result.

Examples

```
# Running formal estimation
data(testdata)
med_model=glm(med~exp+C1+C2+C3, data=testdata, family=binomial) # Fitting mediator's model
out_model=lm(out~med*exp+C1+C2+C3, data=testdata) # Fitting outcome's model
r1 = FormalEstmed (med_model=med_model, out_model=out_model,
data=testdata, exposure = "exp") # Running formal estimation via bootstrapping

# Summary examples
```

```
summary(r1) # Summary of the default settings (Short form and on RD scales).
summary(r1, "long") # Summary of long form and on RD scales.
summary(r1, "long", "OR") # Summary of long form and on OR scales.
summary(r1, "long", "RR") # Summary of long form and on RR scales.
summary(r1, "long", "all") # Summary of long form and on all scales.
summary(r1, form="short", scale="all") # Summary of short form and on all scales.
```

`testdata` *Test Dataset for Causal Mediation Analysis*

Description

A simulated dataset used in examples and vignettes to demonstrate the functionality of the `FormalEstmed` function. Contains variables for exposure (`exp`), mediator (`med`), outcome (`out`), and covariates (`C1`, `C2`, `C3`).

Usage

```
testdata
```

Format

A data frame with 1000 rows and 9 variables:

exp Exposure variable (continuous).

exp2 Exposure variable (binary: 0/1).

exp3 Exposure variable (binary character: male and female).

med Mediator variable (binary: 0/1).

out Outcome variable (continuous).

out2 Outcome variable (ordinal).

C1 Covariate 1 (binary).

C2 Covariate 2 (discrete).

C3 Covariate 3 (continuous).

Source

Simulated data for package demonstration.

Examples

```
data(testdata)
head(testdata)
summary(testdata)
```

um.test1 *Test of Mediation Effects Within One Single Object*

Description

This function tests the difference of various types of effects within one single estimation object from function FormalEstmed. It is used to compare whether an effect is distinct from another.

Usage

```
um.test1 (obj, scale = "RD",type = c("PNDE", "TNIE"), Cf_lv=0.95, verbose=TRUE)
```

Arguments

obj	a resulting object of class "unvs.med" from function FormalEstmed.
scale	a character variable of the effect scales. It can be "RD", "OR" or "RR". The default is "RD".
type	a character variable of the effect types, containing more than two. The default is c("PNDE", "TNIE").
Cf_lv	a numeric variable of the confident interval. The default is 0.95.
verbose	a logical value indicating whether the output is display. The default is TRUE. This is a standard parameter required by CRAN.

Value

No return value, called for displaying the output of the test result.

Examples

```
# Running formal estimation
data(testdata)
med_model=glm(med~exp+C1+C2+C3, data=testdata, family=binomial) # Fitting mediator's model
out_model=lm(out~med*exp+C1+C2+C3, data=testdata) # Fitting outcome's model
r1 = FormalEstmed (med_model=med_model, out_model=out_model,
data=testdata, exposure = "exp") # Running formal estimation

# Test examples
um.test1(r1) # Test of the default settings (PNDE v.s. TNIE on RD scales).
um.test1(r1,"OR") # Test of PNDE v.s. TNIE on OR scales.
# Test of PNDE v.s. TNIE on RD and OR scales:
um.test1(r1,c("RD", "OR"))
# Test of PNDE v.s. TNIE v.s. TE on RD and OR scales:
um.test1(r1,c("RD", "OR"), c("PNDE", "TNIE", "TE"))
# Test of PNDE v.s. TNIE v.s. TE on RD, OR and RR scales:
um.test1(r1, scale=c("RD", "OR", "RR"), type=c("PNDE", "TNIE", "TE"))
```

um.test2 *Test of Mediation Effects Between Two Objects*

Description

This function tests the difference of various types of effects between two estimation objects from function FormalEstmed. It is used to compare whether an effect is distinct from another between two different estimation settings.

Usage

```
um.test2 (obj1, obj2, scale = "RD", type = c("PNDE", "TNIE"), Cf_lv=0.95, verbose=TRUE)
```

Arguments

obj1	a resulting object of class "unvs.med" from function FormalEstmed.
obj2	a resulting object of class "unvs.med" from function FormalEstmed.
scale	a character variable of the effect scales. It can be "RD", "OR" or "RR". The default is "RD".
type	a character variable of the effect types, containing more than two. The default is c("PNDE", "TNIE").
Cf_lv	a numeric variable of the confident interval. The default is 0.95.
verbose	a logical value indicating whether the output is display. The default is TRUE. This is a standard parameter required by CRAN.

Value

No return value, called for displaying the output of the test result.

Examples

```
# Running formal estimation
data(testdata)
med_model=glm(med~exp+exp*C1+C2+C3, data=testdata, family=binomial) # Fitting mediator's model
out_model=lm(out~med*exp+exp*C1+C2+C3, data=testdata) # Fitting outcome's model
r1 = FormalEstmed (med_model=med_model, out_model=out_model,
data=testdata, exposure = "exp", cov_val="C1==1") # Conditioning on C1=1
r0 = FormalEstmed (med_model=med_model, out_model=out_model,
data=testdata, exposure = "exp", cov_val="C1==0") # Conditioning on C1=0

# Test examples
um.test2(r1,r0) # Test of the default settings (PNDE v.s. TNIE on RD scales).
um.test2(r1,r0, c("OR", "RR")) # Test of PNDE v.s. TNIE on OR and RR scales.
# Test of PNDE v.s. TNIE v.s. TE on OR and RR scales:
um.test2(r1,r0, c("OR", "RR"), c("PNDE", "TNIE", "TE"))
# Test of PNDE v.s. TNIE v.s. TE on OR and RR scales with 90% CI:
um.test2(r1,r0, c("OR", "RR"), c("PNDE", "TNIE", "TE"), Cf_lv=0.9)
```

Index

∗ **datasets**
testdata, 17

BootEstimation_for, 2, *14*
BootEstimation_MT, 3, *14*

cond_cov, 4
confirmingX, 5

FormalEstmed, *3–5*, 5, *10, 11*, *15, 16*, *18, 19*

ident_M_type, 9
ident_Y_type, 10

plot.unvs.med, *6*, 11
potentialoutcome_facX, 12, *13*
potentialoutcome_numX, 13

SingleEstimation, *2, 3*, *12, 13*, 14
Statistics, *2, 3*, 15
summary.unvs.med, *6*, 16

testdata, 17

um.test1, *7*, 18
um.test2, *7*, 19