Package ‘sensemakr’

April 28, 2020

Type Package

Title Sensitivity Analysis Tools for Regression Models

Date 2020-04-28

Version 0.1.3

Author Carlos Cinelli [aut, cre],
Jeremy Ferwerda [aut],
Chad Hazlett [aut],
Aaron Rudkin [ctb]

Maintainer Carlos Cinelli <carloscinelli@hotmail.com>

Description Implements a suite of sensitivity analysis tools that extends the traditional omitted variable bias framework and makes it easier to understand the impact of omitted variables in regression models, as discussed in Cinelli, C. and Hazlett, C. (2020), ``Making Sense of Sensitivity: Extending Omitted Variable Bias." Journal of the Royal Statistical Society, Series B (Statistical Methodology) <doi:10.1111/rssb.12348>.

URL https://github.com/chadhazlett/sensemakr

BugReports https://github.com/chadhazlett/sensemakr/issues

License GPL-3

Depends R (>= 3.1.0)

Encoding UTF-8

RoxygenNote 7.0.2

Suggests testthat, covr, knitr, rmarkdown, png, stargazer

LazyData true

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2020-04-28 21:30:03 UTC

R topics documented:

`sensemakr-package` *Sensemakr: extending omitted variable bias*

Description

The sensemakr package implements a suite of sensitivity analysis tools that makes it easier to understand the impact of omitted variables in linear regression models, as discussed in Cinelli and Hazlett (2020).

Details

The main function of the package is `sensemakr`, which computes the most common sensitivity analysis results. After running `sensemakr` you may directly use the plot and print methods in the returned object.

You may also use the other sensitivity functions of the package directly, such as the functions for sensitivity plots (`ovb_contour_plot`, `ovb_extreme_plot`) the functions for computing bias-adjusted estimates and t-values (`adjusted_estimate`, `adjusted_t`), the functions for computing the robustness value and partial R2 (`robustness_value`, `partial_r2`), or the functions for bounding the strength of unobserved confounders (`ovb_bounds`), among others.

More information can be found on the help documentation, vignettes and related papers.

References

Cinelli, C. and Hazlett, C. (2020), "Making Sense of Sensitivity: Extending Omitted Variable Bias." Journal of the Royal Statistical Society, Series B (Statistical Methodology).

Examples

```
# loads dataset
data("darfur")

# runs regression model
model <- lm(peacefactor ~ directlyharmed + age + farmer_dar + herder_dar +
              pastvoted + hhsize_darfur + female + village, data = darfur)

# runs sensemakr for sensitivity analysis
sensitivity <- sensemakr(model, treatment = "directlyharmed",
                         benchmark_covariates = "female",
                         kd = 1:3)
# short description of results
sensitivity

# long description of results
summary(sensitivity)

# plot bias contour of point estimate
plot(sensitivity)

# plot bias contour of t-value
plot(sensitivity, sensitivity.of = "t-value")

# plot extreme scenario
plot(sensitivity, type = "extreme")

# latex code for sensitivity table
ovb_minimal_reporting(sensitivity)

# data.frame with sensitivity statistics
sensitivity$sensitivity_stats

#  data.frame with bounds on the strengh of confounders
sensitivity$bounds

### Using sensitivity functions directly ###

# robustness value of directly harmed q = 1 (reduce estimate to zero)
robustness_value(model, covariates = "directlyharmed")

# robustness value of directly harmed q = 1/2 (reduce estimate in half)
robustness_value(model, covariates = "directlyharmed", q = 1/2)

# robustness value of directly harmed q = 1/2, alpha = 0.05
robustness_value(model, covariates = "directlyharmed", q = 1/2, alpha = 0.05)

# partial R2 of directly harmed with peacefactor
partial_r2(model, covariates = "directlyharmed")

# partial R2 of female with peacefactor
```

```
partial_r2(model, covariates = "female")

# data.frame with sensitivity statistics
sensitivity_stats(model, treatment = "directlyharmed")

# bounds on the strength of confounders using female and age
ovb_bounds(model,
           treatment = "directlyharmed",
           benchmark_covariates = c("female", "age"),
           kd = 1:3)

# adjusted estimate given hypothetical strength of confounder
adjusted_estimate(model, treatment = "directlyharmed", r2dz.x = 0.1, r2yz.dx = 0.1)

# adjusted t-value given hypothetical strength of confounder
adjusted_t(model, treatment = "directlyharmed", r2dz.x = 0.1, r2yz.dx = 0.1)

# bias contour plot directly from lm model
ovb_contour_plot(model,
                 treatment = "directlyharmed",
                 benchmark_covariates = "female",
                 kd = 1:3)

# extreme scenario plot directly from lm model
ovb_extreme_plot(model,
                 treatment = "directlyharmed",
                 benchmark_covariates = "female",
                 kd = 1:3, lim = 0.05)
```

---

add_bound_to_contour *Add bounds to contour plot of omitted variable bias*

Description

Convenience function to add bounds on a sensitivity contour plot created with ovb_contour_plot.

Usage

```
add_bound_to_contour(...)

## S3 method for class 'lm'
add_bound_to_contour(
  model,
  benchmark_covariates,
  kd = 1,
  ky = kd,
  bound_label = NULL,
  treatment = plot.env$treatment,
  reduce = plot.env$reduce,
```

```
  sensitivity.of = plot.env$sensitivity.of,
  label.text = TRUE,
  cex.label.text = 0.7,
  label.bump.x = plot.env$lim * (1/15),
  label.bump.y = plot.env$lim.y * (1/15),
  round = 2,
  ...
)

## S3 method for class 'numeric'
add_bound_to_contour(
  r2dz.x,
  r2yz.dx,
  bound_value = NULL,
  bound_label = NULL,
  label.text = TRUE,
  cex.label.text = 0.7,
  label.bump.x = plot.env$lim * (1/15),
  label.bump.y = plot.env$lim.y * (1/15),
  round = 2,
  ...
)
```

Arguments

`...`	arguments passed to other methods.
`model`	An `lm` object with the outcome regression.
`benchmark_covariates`	The user has two options: (i) character vector of the names of covariates that will be used to bound the plausible strength of the unobserved confounders. Each variable will be considered separately; (ii) a named list with character vector names of covariates that will be used, *as a group*, to bound the plausible strength of the unobserved confounders. The names of the list will be used for the benchmark labels. Note: for factor variables with more than two levels, you need to provide the name of each level as encoded in the `lm` model (the columns of `model.matrix`).
`kd`	numeric vector. Parameterizes how many times stronger the confounder is related to the treatment in comparison to the observed benchmark covariate. Default value is `1` (confounder is as strong as benchmark covariate).
`ky`	numeric vector. Parameterizes how many times stronger the confounder is related to the outcome in comparison to the observed benchmark covariate. Default value is the same as `kd`.
`bound_label`	label to bounds provided manually in `r2dz.x` and `r2yz.dx`.
`treatment`	A character vector with the name of the treatment variable of the model.
`reduce`	Should the bias adjustment reduce or increase the absolute value of the estimated coefficient? Default is `TRUE`.

sensitivity.of	should the contour plot show adjusted estimates ("estimate") or adjusted t-values ("t-value")?
label.text	should label texts be plotted? Default is TRUE.
cex.label.text	size of the label text.
label.bump.x	bump on the x coordinate of label text.
label.bump.y	bump on the y coordinate of label text.
round	integer indicating the number of decimal places to be used for rounding.
r2dz.x	Hypothetical partial R2 of unobserved confounder Z with treatment D, given covariates X.
r2yz.dx	Hypothetical partial R2 of unobserved confounder Z with outcome Y, given covariates X and treatment D.
bound_value	value to be printed in label bound.

Value

The function adds bounds in an existing contour plot and returns 'NULL'.

Examples

```
# runs regression model
model <- lm(peacefactor ~ directlyharmed + age + farmer_dar + herder_dar +
                         pastvoted + hhsize_darfur + female + village,
                         data = darfur)
# contour plot
ovb_contour_plot(model = model, treatment = "directlyharmed")

# add bound 3/1 times stronger than female
add_bound_to_contour(model = model,
                     benchmark_covariates = "female",
                     kd = 3, ky = 1)

# add bound 50/2 times stronger than age
add_bound_to_contour(model = model,
                     benchmark_covariates = "age",
                     kd = 50, ky = 2)
```

---

adjusted_estimate *Bias-adjusted estimates, standard-errors and t-values*

Description

These functions compute bias adjusted estimates (adjusted_estimate), standard-errors (adjusted_se) and t-values (adjusted_t), given a hypothetical strength of the confounder in the partial R2 parameterization.

The functions work either with an lm object, or directly passing in numerical inputs, such as the current coefficient estimate, standard error and degrees of freedom.

Usage

```
adjusted_estimate(...)

## S3 method for class 'lm'
adjusted_estimate(model, treatment, r2dz.x, r2yz.dx, reduce = TRUE, ...)

## S3 method for class 'numeric'
adjusted_estimate(estimate, se, dof, r2dz.x, r2yz.dx, reduce = TRUE, ...)

adjusted_se(...)

## S3 method for class 'numeric'
adjusted_se(se, dof, r2dz.x, r2yz.dx, ...)

## S3 method for class 'lm'
adjusted_se(model, treatment, r2dz.x, r2yz.dx, ...)

adjusted_t(...)

## S3 method for class 'lm'
adjusted_t(model, treatment, r2dz.x, r2yz.dx, reduce = TRUE, h0 = 0, ...)

## S3 method for class 'numeric'
adjusted_t(estimate, se, dof, r2dz.x, r2yz.dx, reduce = TRUE, h0 = 0, ...)

adjusted_partial_r2(...)

## S3 method for class 'numeric'
adjusted_partial_r2(
  estimate,
  se,
  dof,
  r2dz.x,
  r2yz.dx,
  reduce = TRUE,
  h0 = 0,
  ...
)

## S3 method for class 'lm'
adjusted_partial_r2(
  model,
  treatment,
  r2dz.x,
  r2yz.dx,
  reduce = TRUE,
  h0 = 0,
  ...
```

```
)

bias(...)

## S3 method for class 'numeric'
bias(se, dof, r2dz.x, r2yz.dx, ...)

## S3 method for class 'lm'
bias(model, treatment, r2dz.x, r2yz.dx, ...)

relative_bias(...)

## S3 method for class 'lm'
relative_bias(model, treatment, r2dz.x, r2yz.dx, ...)

## S3 method for class 'numeric'
relative_bias(estimate, se, dof, r2dz.x, r2yz.dx, ...)

rel_bias(r.est, est)
```

Arguments

`...`	Arguments passed to other methods. First argument should either be an `lm` model with the outcome regression or a numeric vector with the coefficient estimate.
`model`	An `lm` object with the outcome regression.
`treatment`	A character vector with the name of the treatment variable of the model.
`r2dz.x`	Hypothetical partial R2 of unobserved confounder Z with treatment D, given covariates X.
`r2yz.dx`	Hypothetical partial R2 of unobserved confounder Z with outcome Y, given covariates X and treatment D.
`reduce`	Should the bias adjustment reduce or increase the absolute value of the estimated coefficient? Default is `TRUE`.
`estimate`	Coefficient estimate.
`se`	Standard error of the coefficient estimate.
`dof`	Residual degrees of freedom of the regression.
`h0`	Null hypothesis for computation of the t-value. Default is zero.
`r.est`	restricted estimate. A numerical vector.
`est`	unrestricted estimate. A numerical vector.

Value

Numeric vector with bias, adjusted estimate, standard error, or t-value.

References

Cinelli, C. and Hazlett, C. (2020), "Making Sense of Sensitivity: Extending Omitted Variable Bias." Journal of the Royal Statistical Society, Series B (Statistical Methodology).

Examples

```
# loads data
data("darfur")

# fits model
model <- lm(peacefactor ~ directlyharmed + age +
                          farmer_dar + herder_dar +
                           pastvoted + hhsize_darfur +
                           female + village, data = darfur)

# computes adjusted estimate for confounder with  r2dz.x = 0.05, r2yz.dx = 0.05
adjusted_estimate(model, treatment = "directlyharmed", r2dz.x = 0.05, r2yz.dx = 0.05)

# computes adjusted SE for confounder with  r2dz.x = 0.05, r2yz.dx = 0.05
adjusted_se(model, treatment = "directlyharmed", r2dz.x = 0.05, r2yz.dx = 0.05)

# computes adjusted t-value for confounder with  r2dz.x = 0.05, r2yz.dx = 0.05
adjusted_t(model, treatment = "directlyharmed", r2dz.x = 0.05, r2yz.dx = 0.05)

# Alternatively, pass in numerical values directly.
adjusted_estimate(estimate = 0.09731582, se = 0.02325654,
                  dof = 783, r2dz.x = 0.05, r2yz.dx = 0.05)

adjusted_se(estimate = 0.09731582, se = 0.02325654,
            dof = 783, r2dz.x = 0.05, r2yz.dx = 0.05)

adjusted_t(estimate = 0.09731582, se = 0.02325654,
           dof = 783, r2dz.x = 0.05, r2yz.dx = 0.05)
```

---

colombia — *Data from the 2016 referendum for peace with the FARC in Colombia.*

Description

Data on support for the 2016 referendum for peace with the FARC in Colombia, as discussed in Hazlett and Parente (2020). The main "treatment" variables are `santos2014`, which indicates the share of town population voting in support of Santos in the 2014 Presidential election, and `fat_2011to2015_gtd`, which indicates the number of fatalities due to FARC violence between 2011 and 2015, again at the town level. The main outcome of interest is `yes_vote`, the proportion (0-100) at the town-level voting in support of the peace referendum. The question of interest in Hazlett and Parente (2020) is what can be said about the causal effect of either violence (fatalities) or of political affiliation with Santos, recognizing that analyses of either cannot likely rule out all confounding.

Usage

```
colombia
```

Format

A data frame with 1123 rows and 16 columns.

department Name for the provincial level unit, called departments or departmentos, of which there are 32 in the country.

dept_code Short code for the department

town Name for the town, which is the smallest electoral unit available and is the unit of analysis.

town_code Code for the town.

total_eligible Total eligible voters in the town

yes_vote Proportion (out of 100) voting in favor of the peace deal.

santos10 Proportion (out of 100) voting for Santos in 2010 presidential election.

santos14 Proportion (out of 100) voting for Santos in the 2014 presidential election.

gdppc The town-level GDP per capita.

pop13 Town-level population in 2013.

elev Town's mean elevation.

fat_all Sum of all known fatalities due to FARC violence in the town (from Global Terrorism Database, GTD).

fat_2001to2005_gtd Sum of fatalities due to FARC in the town in 2001 to 2005 (from GTD).

fat_2006to2010_gtd Sum of fatalities due to FARC in the town in 2006 to 2010 (from GTD).

fat_2011to2015_gtd Sum of fatalities due to FARC in the town in 2011 to 2015 (from GTD).

fat_2010to2013 Sum of fatalities due to FARC in the town in 2010 to 2013 (from GTD).

References

Hazlett, C., and Parente, F. (2020). "Who supports peace with the FARC? A sensitivity-based approach under imperfect identification"

Examples

```
# loads data
data(colombia)

#-------------------------------------------------------
# Violence Models
#-------------------------------------------------------

### Model 1 (bivariate)
model1 <- lm(yes_vote ~ fat_2001to2005_gtd, data = colombia)

### Model 2 (more controls, and lagged violence.)
model2 <- lm(yes_vote ~ fat_2001to2005_gtd + fat_2006to2010_gtd +
```

```
                 fat_2011to2015_gtd + total_eligible + santos10 + gdppc ,
               data = colombia)

### Sensitivity analysis - Model 2, for effect of most recent violence.
sense.model2 <- sensemakr(model2,
                          treatment = "fat_2011to2015_gtd",
                          benchmark = "santos10",
                          kd = 1)

### contour plot point estimate
plot(sense.model2)

### contour plot t-value
plot(sense.model2, sensitivity.of = "t-value")

#---------------------------------------------
# Political Affiliation Model
#---------------------------------------------

### Model 3: santos2014 as measure of political support for Santos, with control variables.
model3  <- lm(yes_vote ~ santos14 + fat_2010to2013 + elev + gdppc + pop13,
              data = colombia)

### Sensitivity analysis - Model 3
sense.model3 <- sensemakr(model3, treatment = "santos14",
                          benchmark = c("gdppc","elev"),
                          kd = 3)
summary(sense.model3)

### contour plot point estimate
plot(sense.model3, lim = .9)

### contour plot t-value
plot(sense.model3, sensitivity.of = "t-value", lim = 0.9)
```

`darfur` *Data from survey of Darfurian refugees in eastern Chad.*

Description

Data on attitudes of Darfurian refugees in eastern Chad. The main "treatment" variable is `directlyharmed`, which indicates that the individual was physically injured during attacks on villages in Darfur, largely between 2003 and 2004. The main outcome of interest is `peacefactor`, a measure of pro-peace attitudes.

Key covariates include `herder_dar` (whether they were a herder in Darfur), `farmer_dar` (whether they were a farmer in Darfur), `age`, `female` (indicator for female), and `past_voted` (whether they report having voted in an earlier election, prior to the conflict).

Usage

```
darfur
```

Format

A data frame with 1276 rows and 14 columns.

wouldvote If elections were held in Darfur in the future, would you vote? (0/1)

peacefactor A measure of pro-peace attitudes, from a factor analysis of several questions. Rescaled such that 0 is minimally pro-peace and 1 is maximally pro-peace.

peace_formerenemies Would you be willing to make peace with your former enemies? (0/1)

peace_jjindiv Would you be willing to make peace with Janjweed individuals who carried out violence? (0/1)

peace_jjtribes Would you be willing to make peace with the tribes that were part of the Janjaweed? (0/1)

gos_soldier_execute Should Government of Sudan soldiers who perpetrated attacks on civilians be executed? (0/1)

directlyharmed A binary variable indicating whether the respondent was personally physically injured during attacks on villages in Darfur largely between 2003-2004. 529 respondents report being personally injured, while 747 do not report being injured.

age Age of respondent in whole integer years. Ages in the data range from 18 to 100.

farmer_dar The respondent was a farmer in Darfur (0/1). 1,051 respondents were farmers, 225 were not.

herder_dar The respondent was a herder in Darfur (0/1). 190 respondents were farmers, 1,086 were not.

pastvoted The respondent reported having voted in a previous election before the conflict (0/1). 821 respondents reported having voted in a previous election, 455 reported not having voted in a previous election.

hhsize_darfur Household size while in Darfur.

village Factor variable indicating village of respondent. 486 unique villages are accounted for in the data.

female The respondent identifies as female (0/1). 582 respondents are female-identified, 694 are not.

References

Cinelli, C. and Hazlett, C. (2020), "Making Sense of Sensitivity: Extending Omitted Variable Bias." Journal of the Royal Statistical Society, Series B (Statistical Methodology).

Hazlett, Chad. (2019) "Angry or Weary? How Violence Impacts Attitudes toward Peace among Darfurian Refugees." Journal of Conflict Resolution: 0022002719879217.

group_partial_r2 *Partial R2 of groups of covariates in a linear regression model*

Description

This function computes the partial R2 of a group of covariates in a linear regression model.

Usage

```
group_partial_r2(...)

## S3 method for class 'lm'
group_partial_r2(model, covariates, ...)

## S3 method for class 'numeric'
group_partial_r2(F.stats, p, dof, ...)
```

Arguments

`...`	arguments passed to other methods. First argument should either be an `lm` object with the regression model or a numeric vector with the F-statistics for the group of covariates.
`model`	an `lm` object with the regression model
`covariates`	model covariates for which their grouped partial R2 will be computed.
`F.stats`	F-statistics for the group of covariates.
`p`	number of parameters in the model.
`dof`	residual degrees of freedom of the model.

Value

A numeric vector with the computed partial R2.

Examples

```
data("darfur")

model <- lm(peacefactor ~ directlyharmed + age + farmer_dar + herder_dar +
             pastvoted + hhsize_darfur + female + village, data = darfur)

group_partial_r2(model, covariates = c("female", "pastvoted"))
```

`model_helper` *Helper function for extracting model statistics*

Description

This is an internal function used for extracting the necessary statistics from the models.

Usage

```
model_helper(model, covariates = NULL, ...)
```

Arguments

`model`	model to extract statistics from
`covariates`	model covariates from which statistics will be extracted.
`...`	arguments passed to other methods.

`ovb_bounds` *Bounds on the strength of unobserved confounders using observed covariates*

Description

Bounds on the strength of unobserved confounders using observed covariates, as in Cinelli and Hazlett (2020). The main generic function is `ovb_bounds`, which can compute both the bounds on the strength of confounding as well as the adjusted estimates, standard errors, t-values and confidence intervals.

Other functions that compute only the bounds on the strength of confounding are also provided. These functions may be useful when computing benchmarks for using only summary statistics from papers you see in print.

Usage

```
ovb_bounds(...)

## S3 method for class 'lm'
ovb_bounds(
  model,
  treatment,
  benchmark_covariates = NULL,
  kd = 1,
  ky = kd,
  reduce = TRUE,
  bound = c("partial r2", "partial r2 no D", "total r2"),
  adjusted_estimates = TRUE,
```

```
  alpha = 0.05,
  h0 = 0,
  ...
)

ovb_partial_r2_bound(...)

## S3 method for class 'numeric'
ovb_partial_r2_bound(
  r2dxj.x,
  r2yxj.dx,
  kd = 1,
  ky = kd,
  bound_label = "manual",
  ...
)
```

Arguments

`...` arguments passed to other methods. First argument should either be an `lm` model with the outcome regression, or a `formula` describing the model along with the `data.frame` containing the variables of the model.

`model` An `lm` object with the outcome regression.

`treatment` A character vector with the name of the treatment variable of the model.

`benchmark_covariates` The user has two options: (i) character vector of the names of covariates that will be used to bound the plausible strength of the unobserved confounders. Each variable will be considered separately; (ii) a named list with character vector names of covariates that will be used, *as a group*, to bound the plausible strength of the unobserved confounders. The names of the list will be used for the benchmark labels. Note: for factor variables with more than two levels, you need to provide the name of each level as encoded in the `lm` model (the columns of `model.matrix`).

`kd` numeric vector. Parameterizes how many times stronger the confounder is related to the treatment in comparison to the observed benchmark covariate. Default value is `1` (confounder is as strong as benchmark covariate).

`ky` numeric vector. Parameterizes how many times stronger the confounder is related to the outcome in comparison to the observed benchmark covariate. Default value is the same as `kd`.

`reduce` Should the bias adjustment reduce or increase the absolute value of the estimated coefficient? Default is `TRUE`.

`bound` type of bounding procedure. Currently only `"partial r2"` is implemented.

`adjusted_estimates` should the bounder also compute the adjusted estimates? Default is `TRUE`.

`alpha` significance level for computing the adjusted confidence intervals. Default is 0.05.

h0	Null hypothesis for computation of the t-value. Default is zero.
r2dxj.x	partial R2 of covariate Xj with the treatment D (after partialling out the effect of the remaining covariates X, excluding Xj).
r2yxj.dx	partial R2 of covariate Xj with the outcome Y (after partialling out the effect of the remaining covariates X, excluding Xj).
bound_label	label to bounds provided manually in r2dz.x and r2yz.dx.

Details

Currently it implements only the bounds based on partial R2. Other bounds will be implemented soon.

Value

The function ovb_bounds returns a data.frame with the bounds on the strength of the unobserved confounder as well with the adjusted point estimates, standard errors and t-values (optional, controlled by argument adjusted_estimates = TRUE).

The function 'ovb_partial_r2_bound()' returns only data.frame with the bounds on the strength of the unobserved confounder. Adjusted estimates, standard errors and t-values (among other quantities) need to be computed manually by the user using those bounds with the functions adjusted_estimate, adjusted_se and adjusted_t.

References

Cinelli, C. and Hazlett, C. (2020), "Making Sense of Sensitivity: Extending Omitted Variable Bias." Journal of the Royal Statistical Society, Series B (Statistical Methodology).

Cinelli, C. and Hazlett, C. (2020), "Making Sense of Sensitivity: Extending Omitted Variable Bias." Journal of the Royal Statistical Society, Series B (Statistical Methodology).

Examples

```
# runs regression model
model <- lm(peacefactor ~ directlyharmed + age + farmer_dar + herder_dar +
             pastvoted + hhsize_darfur + female + village, data = darfur)

# bounds on the strength of confounders 1, 2, or 3 times as strong as female
# and 1,2, or 3 times as strong as pastvoted
ovb_bounds(model, treatment = "directlyharmed",
          benchmark_covariates = c("female", "pastvoted"),
          kd = 1:3)


#########################################################
## Let's construct bounds from summary statistics only ##
#########################################################
# Suppose you didn't have access to the data, but only to
# the treatment and outcome regression tables.
# You can still compute the bounds.
```

```
# Use the t statistic of female in the outcome regression
# to compute the partial R2 of female with the outcome.
r2yxj.dx <- partial_r2(t_statistic = -9.789, dof = 783)

# Use the t-value of female in the *treatment* regression
# to compute the partial R2 of female with the treatment
r2dxj.x <- partial_r2(t_statistic = -2.680, dof = 783)

# Compute manually bounds on the strength of confounders 1, 2, or 3
# times as strong as female
bounds <- ovb_partial_r2_bound(r2dxj.x = r2dxj.x,
                               r2yxj.dx = r2yxj.dx,
                               kd = 1:3,
                               ky = 1:3,
                               bound_label = paste(1:3, "x", "female"))
# Compute manually adjusted estimates
bound.values <- adjusted_estimate(estimate = 0.0973,
                                  se = 0.0232,
                                  dof = 783,
                                  r2dz.x = bounds$r2dz.x,
                                  r2yz.dx = bounds$r2yz.dx)

# Plot contours and bounds
ovb_contour_plot(estimate = 0.0973,
                 se = 0.0232,
                 dof = 783)
add_bound_to_contour(bounds, bound_value = bound.values)
```

`ovb_contour_plot` *Contour plots of omitted variable bias*

Description

Contour plots of omitted variable bias for sensitivity analysis. The main inputs are an `lm` model, the treatment variable and the covariates used for benchmarking the strength of unobserved confounding.

The horizontal axis of the plot shows hypothetical values of the partial R2 of the unobserved confounder(s) with the treatment. The vertical axis shows hypothetical values of the partial R2 of the unobserved confounder(s) with the outcome. The contour levels represent the adjusted estimates (or t-values) of the treatment effect. The reference points are the bounds on the partial R2 of the unobserved confounder if it were `k` times "as strong" as the observed covariate used for benchmarking (see arguments `kd` and `ky`). The dotted red line show the chosen critical threshold (for instance, zero): confounders with such strength (or stronger) are sufficient to invalidate the research conclusions. All results are exact for single confounders and conservative for multiple/nonlinear confounders.

See Cinelli and Hazlett (2020) for details.

Usage

```
ovb_contour_plot(...)

## S3 method for class 'lm'
ovb_contour_plot(
  model,
  treatment,
  benchmark_covariates = NULL,
  kd = 1,
  ky = kd,
  r2dz.x = NULL,
  r2yz.dx = r2dz.x,
  bound_label = "manual",
  sensitivity.of = c("estimate", "t-value"),
  reduce = TRUE,
  estimate.threshold = 0,
  t.threshold = 2,
  nlevels = 10,
  col.contour = "grey40",
  col.thr.line = "red",
  label.text = TRUE,
  cex.label.text = 0.7,
  round = 3,
  ...
)

## S3 method for class 'formula'
ovb_contour_plot(
  formula,
  data,
  treatment,
  benchmark_covariates = NULL,
  kd = 1,
  ky = kd,
  r2dz.x = NULL,
  r2yz.dx = r2dz.x,
  bound_label = NULL,
  sensitivity.of = c("estimate", "t-value"),
  reduce = TRUE,
  estimate.threshold = 0,
  t.threshold = 2,
  nlevels = 10,
  col.contour = "grey40",
  col.thr.line = "red",
  label.text = TRUE,
  cex.label.text = 0.7,
  round = 3,
  ...
```

```
)

## S3 method for class 'numeric'
ovb_contour_plot(
  estimate,
  se,
  dof,
  r2dz.x = NULL,
  r2yz.dx = r2dz.x,
  bound_label = rep("manual", length(r2dz.x)),
  sensitivity.of = c("estimate", "t-value"),
  reduce = TRUE,
  estimate.threshold = 0,
  t.threshold = 2,
  lim = NULL,
  lim.y = NULL,
  nlevels = 10,
  col.contour = "black",
  col.thr.line = "red",
  label.text = TRUE,
  cex.label.text = 0.7,
  label.bump.x = NULL,
  label.bump.y = NULL,
  xlab = NULL,
  ylab = NULL,
  cex.lab = 0.8,
  cex.axis = 0.8,
  cex.main = 1,
  asp = lim/lim.y,
  list.par = list(mar = c(4, 4, 1, 1), pty = "s"),
  round = 3,
  ...
)
```

Arguments

`...` arguments passed to other methods. First argument should either be an `lm` model with the outcome regression, a `formula` describing the model along with the `data.frame` containing the variables of the model, or a numeric vector with the coefficient estimate.

`model` An `lm` object with the outcome regression.

`treatment` A character vector with the name of the treatment variable of the model.

`benchmark_covariates`
The user has two options: (i) character vector of the names of covariates that will be used to bound the plausible strength of the unobserved confounders. Each variable will be considered separately; (ii) a named list with character vector names of covariates that will be used, *as a group*, to bound the plausible strength of the unobserved confounders. The names of the list will be used for

the benchmark labels. Note: for factor variables with more than two levels, you need to provide the name of each level as encoded in the `lm` model (the columns of `model.matrix`).

`kd` numeric vector. Parameterizes how many times stronger the confounder is related to the treatment in comparison to the observed benchmark covariate. Default value is `1` (confounder is as strong as benchmark covariate).

`ky` numeric vector. Parameterizes how many times stronger the confounder is related to the outcome in comparison to the observed benchmark covariate. Default value is the same as `kd`.

`r2dz.x` Hypothetical partial R2 of unobserved confounder Z with treatment D, given covariates X.

`r2yz.dx` Hypothetical partial R2 of unobserved confounder Z with outcome Y, given covariates X and treatment D.

`bound_label` label to bounds provided manually in `r2dz.x` and `r2yz.dx`.

`sensitivity.of` should the contour plot show adjusted estimates (`"estimate"`) or adjusted t-values (`"t-value"`)?

`reduce` Should the bias adjustment reduce or increase the absolute value of the estimated coefficient? Default is `TRUE`.

`estimate.threshold`
critical threshold for the point estimate.

`t.threshold` critical threshold for the t-value.

`nlevels` number of levels for the contour plot.

`col.contour` color of contour lines.

`col.thr.line` color of threshold contour line.

`label.text` should label texts be plotted? Default is `TRUE`.

`cex.label.text` size of the label text.

`round` number of digits to show in contours and bound values

`formula` an object of the class `formula`: a symbolic description of the model to be fitted.

`data` data needed only when you pass a formula as first parameter. An object of the class `data.frame` containing the variables used in the analysis.

`estimate` Coefficient estimate.

`se` Standard error of the coefficient estimate.

`dof` Residual degrees of freedom of the regression.

`lim` sets limit for x-axis. If ‘NULL‘, limits are computed automatically.

`lim.y` sets limit for y-axis. If ‘NULL‘, limits are computed automatically.

`label.bump.x` bump on the x coordinate of label text.

`label.bump.y` bump on the y coordinate of label text.

`xlab` label of x axis. If ‘NULL‘, default label is used.

`ylab` label of y axis. If ‘NULL‘, default label is used.

`cex.lab` The magnification to be used for x and y labels relative to the current setting of cex.

`cex.axis`	The magnification to be used for axis annotation relative to the current setting of cex.
`cex.main`	The magnification to be used for main titles relative to the current setting of cex.
`asp`	the y/x aspect ratio. Default is 1.
`list.par`	arguments to be passed to `par`. It needs to be a named list.

Value

The function returns invisibly the data used for the contour plot (contour grid and bounds).

References

Cinelli, C. and Hazlett, C. (2020), "Making Sense of Sensitivity: Extending Omitted Variable Bias." Journal of the Royal Statistical Society, Series B (Statistical Methodology).

Examples

```
# runs regression model
model <- lm(peacefactor ~ directlyharmed + age + farmer_dar + herder_dar +
                          pastvoted + hhsize_darfur + female + village,
                          data = darfur)
# contour plot
ovb_contour_plot(model, treatment = "directlyharmed",
                        benchmark_covariates = "female",
                        kd = 1:2)
```

`ovb_extreme_plot` *Extreme scenarios plots of omitted variable bias*

Description

Extreme scenario plots of omitted variable bias for sensitivity analysis. The main inputs are an `lm` model, the treatment variable and the covariates used for benchmarking the strength of unobserved confounding.

The horizontal axis shows the partial R2 of the unobserved confounder(s) with the treatment. The vertical axis shows the adjusted treatment effect estimate. The partial R2 of the confounder with the outcome is represented by *different curves* for each scenario, as given by the parameter `r2yz.dx`. The red marks on horizontal axis are bounds on the partial R2 of the unobserved confounder `kd` times as strong as the covariates used for benchmarking. The dotted red line represent the threshold for the effect estimate deemed to be problematic (for instance, zero).

See Cinelli and Hazlett (2020) for details.

Usage

```
ovb_extreme_plot(...)

## S3 method for class 'lm'
ovb_extreme_plot(
  model,
  treatment,
  benchmark_covariates = NULL,
  kd = 1,
  r2yz.dx = c(1, 0.75, 0.5),
  r2dz.x = NULL,
  reduce = TRUE,
  threshold = 0,
  lim = min(c(r2dz.x + 0.1, 0.5)),
  legend = TRUE,
  cex.legend = 0.65,
  legend.bty = "n",
  ...
)

## S3 method for class 'formula'
ovb_extreme_plot(
  formula,
  data,
  treatment,
  benchmark_covariates = NULL,
  kd = 1,
  r2yz.dx = c(1, 0.75, 0.5),
  r2dz.x = NULL,
  reduce = TRUE,
  threshold = 0,
  lim = min(c(r2dz.x + 0.1, 0.5)),
  legend = TRUE,
  cex.legend = 0.65,
  legend.bty = "n",
  ...
)

## S3 method for class 'numeric'
ovb_extreme_plot(
  estimate,
  se,
  dof,
  r2dz.x = NULL,
  r2yz.dx = c(1, 0.75, 0.5),
  reduce = TRUE,
  threshold = 0,
  lim = min(c(r2dz.x + 0.1, 0.5)),
```

```
  legend = TRUE,
  legend.title = NULL,
  cex.legend = 0.65,
  legend.bty = "n",
  xlab = NULL,
  ylab = NULL,
  cex.lab = 0.7,
  cex.axis = 0.7,
  list.par = list(oma = c(1, 1, 1, 1)),
  ...
)
```

Arguments

`...`	arguments passed to other methods. First argument should either be an `lm` model with the outcome regression, a `formula` describing the model along with the `data.frame` containing the variables of the model, or a numeric vector with the coefficient estimate.
`model`	An `lm` object with the outcome regression.
`treatment`	A character vector with the name of the treatment variable of the model.
`benchmark_covariates`	The user has two options: (i) character vector of the names of covariates that will be used to bound the plausible strength of the unobserved confounders. Each variable will be considered separately; (ii) a named list with character vector names of covariates that will be used, *as a group*, to bound the plausible strength of the unobserved confounders. The names of the list will be used for the benchmark labels. Note: for factor variables with more than two levels, you need to provide the name of each level as encoded in the `lm` model (the columns of `model.matrix`).
`kd`	numeric vector. Parameterizes how many times stronger the confounder is related to the treatment in comparison to the observed benchmark covariate. Default value is `1` (confounder is as strong as benchmark covariate).
`r2yz.dx`	Hypothetical partial R2 of unobserved confounder Z with outcome Y, given covariates X and treatment D.
`r2dz.x`	Hypothetical partial R2 of unobserved confounder Z with treatment D, given covariates X.
`reduce`	Should the bias adjustment reduce or increase the absolute value of the estimated coefficient? Default is `TRUE`.
`threshold`	estimate threshold.
`lim`	sets limit for x-axis. If ‘NULL‘, limits are computed automatically.
`legend`	should legend be plotted? Default is `TRUE`.
`cex.legend`	size of the text for the legend.
`legend.bty`	legend box. See `bty` argument of par.
`formula`	an object of the class `formula`: a symbolic description of the model to be fitted.

data	data needed only when you pass a formula as first parameter. An object of the class data.frame containing the variables used in the analysis.
estimate	Coefficient estimate.
se	Standard error of the coefficient estimate.
dof	Residual degrees of freedom of the regression.
legend.title	the legend title. If NULL, then default legend is used.
xlab	label of x axis. If 'NULL', default label is used.
ylab	label of y axis. If 'NULL', default label is used.
cex.lab	The magnification to be used for x and y labels relative to the current setting of cex.
cex.axis	The magnification to be used for axis annotation relative to the current setting of cex.
list.par	arguments to be passed to par. It needs to be a named list.

Value

The function returns invisibly the data used for the extreme plot.

References

Cinelli, C. and Hazlett, C. (2020), "Making Sense of Sensitivity: Extending Omitted Variable Bias." Journal of the Royal Statistical Society, Series B (Statistical Methodology).

Examples

```
# runs regression model
model <- lm(peacefactor ~ directlyharmed + age + farmer_dar + herder_dar +
                         pastvoted + hhsize_darfur + female + village,
                         data = darfur)
# extreme scenarios plot
ovb_extreme_plot(model, treatment = "directlyharmed",
                        benchmark_covariates = "female",
                        kd = 1:2,
                        lim = 0.05)
```

---

partial_r2 *Computes the partial R2 and partial (Cohen's) f2*

Description

These functions computes the partial R2 and the partial (Cohen's) f2 for a linear regression model. The partial R2 describes how much of the residual variance of the outcome (after partialing out the other covariates) a covariate explains.

The partial R2 can be used as an extreme-scenario sensitivity analysis to omitted variables. Considering an unobserved confounder that explains 100% of the residual variance of the outcome, the partial R2 describes how strongly associated with the treatment this unobserved confounder would need to be in order to explain away the estimated effect. For details see Cinelli and Hazlett (2020).

The partial (Cohen's) f2 is a common measure of effect size (a transformation of the partial R2) that can also be used directly for sensitivity analysis using a bias factor table.

The function `partial_r2` computes the partial R2. The function `partial_f2` computes the partial f2 and the function `partial_f` the partial f. They can take as input an `lm` object or you may pass directly t-value and degrees of freedom.

For partial R2 of groups of covariates, check `group_partial_r2`.

Usage

```
partial_r2(...)

## S3 method for class 'lm'
partial_r2(model, covariates = NULL, ...)

## S3 method for class 'numeric'
partial_r2(t_statistic, dof, ...)

partial_f2(...)

## S3 method for class 'numeric'
partial_f2(t_statistic, dof, ...)

## S3 method for class 'lm'
partial_f2(model, covariates = NULL, ...)

partial_f(...)
```

Arguments

`...`	arguments passed to other methods. First argument should either be an `lm` object with the regression model or a numeric vector with the t-value of the coefficient estimate
`model`	an `lm` object with the regression model
`covariates`	model covariates for which the partial R2 will be computed. Default is to compute the partial R2 of all covariates.
`t_statistic`	`numeric` vector with the t-value of the coefficient estimates
`dof`	residual degrees of freedom of the regression

Value

A numeric vector with the computed partial R2, f2, or f.

References

Cinelli, C. and Hazlett, C. (2020), "Making Sense of Sensitivity: Extending Omitted Variable Bias." Journal of the Royal Statistical Society, Series B (Statistical Methodology).

Examples

```
# using an lm object
## loads data
data("darfur")

## fits model
model <- lm(peacefactor ~ directlyharmed + age + farmer_dar + herder_dar +
             pastvoted + hhsize_darfur + female + village, data = darfur)

## partial R2 of the treatment (directly harmed) with the outcome (peacefactor)
partial_r2(model, covariates = "directlyharmed")

## partial R2 of female with the outcome
partial_r2(model, covariates = "female")

# you can also provide the statistics directly
partial_r2(t_statistic = 4.18445, dof = 783)
```

---

plot.sensemakr — *Sensitivity analysis plots for* sensemakr

Description

This function provides the contour and extreme scenario sensitivity plots of the sensitivity analysis results obtained with the function sensemakr. They are basically dispatchers to the core plot functions ovb_contour_plot and ovb_extreme_plot.

Usage

```
## S3 method for class 'sensemakr'
plot(x, type = c("contour", "extreme"), ...)
```

Arguments

x	an object of class sensemakr.
type	type of sensitivity plot. It can be "contour", for contour plots of omitted variable bias as in ovb_contour_plot; or, "extreme" for extreme scenarios plots of omitted variable bias as in ovb_extreme_plot.

...	arguments passed to the plot functions. Check arguments in ovb_contour_plot and ovb_extreme_plot.

print.sensemakr *Sensitivity analysis print and summary methods for* sensemakr

Description

The print and summary methods provide verbal descriptions of the sensitivity analysis results obtained with the function sensemakr. The function ovb_minimal_reporting provides latex or html code for a minimal sensitivity analysis reporting, as suggested in Cinelli and Hazlett (2020).

Usage

```
## S3 method for class 'sensemakr'
print(x, digits = max(3L, getOption("digits") - 2L), ...)

## S3 method for class 'sensemakr'
summary(object, digits = max(3L, getOption("digits") - 3L), ...)

ovb_minimal_reporting(
  x,
  digits = 3,
  verbose = TRUE,
  format = c("latex", "html", "pure_html"),
  ...
)
```

Arguments

x	an object of class sensemakr.
digits	minimal number of *significant* digits.
...	arguments passed to other methods.
object	an object of class sensemakr.
verbose	if ‘TRUE‘, the function prints the LaTeX code with cat
format	code format to print, either latex or html. The default html version has some mathematical content that requires mathjax or equivalent library to parse. If you need only html, use the option "pure_html".

Value

The function ovb_minimal_reporting returns the LaTeX/HTML code invisibly in character form and also prints with cat the LaTeX code. To suppress automatic printing, set verbose = FALSE.

References

Cinelli, C. and Hazlett, C. (2020), "Making Sense of Sensitivity: Extending Omitted Variable Bias." Journal of the Royal Statistical Society, Series B (Statistical Methodology).

Examples

```
# runs regression model
model <- lm(peacefactor ~ directlyharmed + age + farmer_dar + herder_dar +
                         pastvoted + hhsize_darfur + female + village,
                         data = darfur)

# runs sensemakr for sensitivity analysis
sensitivity <- sensemakr(model, treatment = "directlyharmed",
                               benchmark_covariates = "female",
                               kd = 1:3)
# print
sensitivity

# summary
summary(sensitivity)

# prints latex code for minimal sensitivity analysis reporting
ovb_minimal_reporting(sensitivity)
```

---

resid_maker *Creates orthogonal residuals*

Description

This function is an auxiliary function for simulation purposes. It creates a vector of n standard normal random variables, residualizes this vector against a matrix of covariates C, then standardizes the vector again.

Usage

```
resid_maker(n, C)
```

Arguments

n	sample size.
C	a numeric matrix with n rows and p columns.

Value

The function returns a numeric vector orthogonal to C.

`robustness_value` *Computes the robustness value*

Description

This function computes the robustness value of a regression coefficient.

The robustness value describes the minimum strength of association (parameterized in terms of partial R2) that omitted variables would need to have both with the treatment and with the outcome to change the estimated coefficient by a certain amount (for instance, to bring it down to zero).

For instance, a robustness value of 1% means that an unobserved confounder that explain 1% of the residual variance of the outcome and 1% of the residual variance of the treatment is strong enough to explain away the estimated effect. Whereas a robustness value of 90% means that any unobserved confounder that explain less than 90% of the residual variance of both the outcome and the treatment assignment cannot fully account for the observed effect. You may also compute robustness value taking into account sampling uncertainty. See details in Cinelli and Hazlett (2020).

The function robustness_value can take as input an `lm` object or you may directly pass the t-value and degrees of freedom.

Usage

```
robustness_value(...)

## S3 method for class 'lm'
robustness_value(model, covariates = NULL, q = 1, alpha = 1, ...)

## S3 method for class 'numeric'
robustness_value(t_statistic, dof, q = 1, alpha = 1, ...)
```

Arguments

`...`	arguments passed to other methods. First argument should either be an `lm` model with the regression model or a numeric vector with the t-value of the coefficient estimate
`model`	an `lm` object with the regression model.
`covariates`	model covariates for which the robustness value will be computed. Default is to compute the robustness value of all covariates.
`q`	percent change of the effect estimate that would be deemed problematic. Default is `1`, which means a reduction of 100% of the current effect estimate (bring estimate to zero). It has to be greater than zero.
`alpha`	significance level.
`t_statistic`	`numeric` vector with the t-value of the coefficient estimates
`dof`	residual degrees of freedom of the regression

Value

The function returns a numerical vector with the robustness value. The arguments q and alpha are saved as attributes of the vector for reference.

References

Cinelli, C. and Hazlett, C. (2020), "Making Sense of Sensitivity: Extending Omitted Variable Bias." Journal of the Royal Statistical Society, Series B (Statistical Methodology).

Examples

```
# using an lm object
## loads data
data("darfur")

## fits model
model <- lm(peacefactor ~ directlyharmed + age + farmer_dar + herder_dar +
             pastvoted + hhsize_darfur + female + village, data = darfur)

## robustness value of directly harmed q =1 (reduce estimate to zero)
robustness_value(model, covariates = "directlyharmed")

## robustness value of directly harmed q = 1/2 (reduce estimate in half)
robustness_value(model, covariates = "directlyharmed", q = 1/2)

## robustness value of directly harmed q = 1/2, alpha = 0.05
## (reduce estimate in half, with 95% confidence)
robustness_value(model, covariates = "directlyharmed", q = 1/2, alpha = 0.05)

# you can also provide the statistics directly
robustness_value(t_statistic = 4.18445, dof = 783)
```

---

sensemakr   *Sensitivity analysis to unobserved confounders*

Description

This function performs sensitivity analysis to omitted variables as discussed in Cinelli and Hazlett (2020). It returns an object of class `sensemakr` with several pre-computed sensitivity statistics for reporting. After running `sensemakr` you may directly use the `plot`, `print` and `summary` methods in the returned object.

Usage

```
sensemakr(...)

## S3 method for class 'lm'
sensemakr(
```

```
  model,
  treatment,
  benchmark_covariates = NULL,
  kd = 1,
  ky = kd,
  q = 1,
  alpha = 0.05,
  r2dz.x = NULL,
  r2yz.dx = r2dz.x,
  bound_label = "Manual Bound",
  reduce = TRUE,
  ...
)

## S3 method for class 'formula'
sensemakr(
  formula,
  data,
  treatment,
  benchmark_covariates = NULL,
  kd = 1,
  ky = kd,
  q = 1,
  alpha = 0.05,
  r2dz.x = NULL,
  r2yz.dx = r2dz.x,
  bound_label = "",
  reduce = TRUE,
  ...
)

## S3 method for class 'numeric'
sensemakr(
  estimate,
  se,
  dof,
  treatment = "D",
  q = 1,
  alpha = 0.05,
  r2dz.x = NULL,
  r2yz.dx = r2dz.x,
  bound_label = "manual_bound",
  r2dxj.x = NULL,
  r2yxj.dx = r2dxj.x,
  benchmark_covariates = "manual_benchmark",
  kd = 1,
  ky = kd,
  reduce = TRUE,
```

```
  ...
)
```

Arguments

- `...` arguments passed to other methods. First argument should either be an `lm` model with the outcome regression, or a `formula` describing the model along with the `data.frame` containing the variables of the model.
- `model` An `lm` object with the outcome regression.
- `treatment` A character vector with the name of the treatment variable of the model.
- `benchmark_covariates` The user has two options: (i) character vector of the names of covariates that will be used to bound the plausible strength of the unobserved confounders. Each variable will be considered separately; (ii) a named list with character vector names of covariates that will be used, *as a group*, to bound the plausible strength of the unobserved confounders. The names of the list will be used for the benchmark labels. Note: for factor variables with more than two levels, you need to provide the name of each level as encoded in the `lm` model (the columns of `model.matrix`).
- `kd` numeric vector. Parameterizes how many times stronger the confounder is related to the treatment in comparison to the observed benchmark covariate. Default value is `1` (confounder is as strong as benchmark covariate).
- `ky` numeric vector. Parameterizes how many times stronger the confounder is related to the outcome in comparison to the observed benchmark covariate. Default value is the same as `kd`.
- `q` percent change of the effect estimate that would be deemed problematic. Default is `1`, which means a reduction of 100% of the current effect estimate (bring estimate to zero). It has to be greater than zero.
- `alpha` significance level.
- `r2dz.x` Hypothetical partial R2 of unobserved confounder Z with treatment D, given covariates X.
- `r2yz.dx` Hypothetical partial R2 of unobserved confounder Z with outcome Y, given covariates X and treatment D.
- `bound_label` label to bounds provided manually in `r2dz.x` and `r2yz.dx`.
- `reduce` Should the bias adjustment reduce or increase the absolute value of the estimated coefficient? Default is `TRUE`.
- `formula` an object of the class `formula`: a symbolic description of the model to be fitted.
- `data` data needed only when you pass a formula as first parameter. An object of the class `data.frame` containing the variables used in the analysis.
- `estimate` Coefficient estimate.
- `se` Standard error of the coefficient estimate.
- `dof` Residual degrees of freedom of the regression.
- `r2dxj.x` partial R2 of covariate Xj with the treatment D (after partialling out the effect of the remaining covariates X, excluding Xj).
- `r2yxj.dx` partial R2 of covariate Xj with the outcome Y (after partialling out the effect of the remaining covariates X, excluding Xj).

Value

An object of class sensemakr, containing:

info A data.frame with the general information of the analysis, including the formula used, the name of the treatment variable, parameter values such as q, alpha, and whether the bias is assumed to reduce the current estimate.

sensitivity_stats A data.frame with the sensitivity statistics for the treatment variable, as computed by the function sensitivity_stats.

bounds A data.frame with bounds on the strength of confounding according to some benchmark covariates, as computed by the function ovb_bounds.

References

Cinelli, C. and Hazlett, C. (2020), "Making Sense of Sensitivity: Extending Omitted Variable Bias." Journal of the Royal Statistical Society, Series B (Statistical Methodology).

See Also

The function sensemakr is a convenience function. You may use the other sensitivity functions of the package directly, such as the functions for sensitivity plots (ovb_contour_plot, ovb_extreme_plot) the functions for computing bias-adjusted estimates and t-values (adjusted_estimate, adjusted_t), the functions for computing the robustness value and partial R2 (robustness_value, partial_r2), or the functions for bounding the strength of unobserved confounders (ovb_bounds), among others.

Examples

```
# loads dataset
data("darfur")

# runs regression model
model <- lm(peacefactor ~ directlyharmed + age + farmer_dar + herder_dar +
                         pastvoted + hhsize_darfur + female + village, data = darfur)

# runs sensemakr for sensitivity analysis
sensitivity <- sensemakr(model, treatment = "directlyharmed",
                               benchmark_covariates = "female",
                               kd = 1:3)
# short description of results
sensitivity

# long description of results
summary(sensitivity)

# plot bias contour of point estimate
plot(sensitivity)

# plot bias contour of t-value
plot(sensitivity, sensitivity.of = "t-value")

# plot extreme scenario
```

```
plot(sensitivity, type = "extreme")

# latex code for sensitivity table
ovb_minimal_reporting(sensitivity)
```

---

sensitivity_stats *Sensitivity statistics for regression coefficients*

Description

Convenience function that computes the robustness_value, partial_r2 and partial_f2 of the coefficient of interest.

Usage

```
sensitivity_stats(...)

## S3 method for class 'lm'
sensitivity_stats(model, treatment, q = 1, alpha = 0.05, reduce = TRUE, ...)

## S3 method for class 'numeric'
sensitivity_stats(
  estimate,
  se,
  dof,
  treatment = "treatment",
  q = 1,
  alpha = 0.05,
  reduce = TRUE,
  ...
)
```

Arguments

`...`	Arguments passed to other methods. First argument should either be an `lm` model with the outcome regression or a numeric vector with the coefficient estimate.
`model`	An `lm` object with the outcome regression.
`treatment`	A character vector with the name of the treatment variable of the model.
`q`	percent change of the effect estimate that would be deemed problematic. Default is `1`, which means a reduction of 100% of the current effect estimate (bring estimate to zero). It has to be greater than zero.
`alpha`	significance level.
`reduce`	Should the bias adjustment reduce or increase the absolute value of the estimated coefficient? Default is `TRUE`.

`estimate`	Coefficient estimate.
`se`	Standard error of the coefficient estimate.
`dof`	Residual degrees of freedom of the regression.

Value

A `data.frame` containing the following quantities:

treatment a character with the name of the treatment variable

estimate a numeric vector with the estimated effect of the treatment

se a numeric vector with the estimated standard error of the treatment effect

t_statistics a numeric vector with the t-value of the treatment

r2yd.x a numeric vector with the partial R2 of the treatment and the outcome, see details in `partial_r2`

rv_q a numeric vector with the robustness value of the treatment, see details in `robustness_value`

rv_qa a numeric vector with the robustness value of the treatment considering statistical significance, see details in `robustness_value`

f2yd.x a numeric vector with the partial (Cohen's) f2 of the treatment with the outcome, see details in `partial_f2`

dof a numeric vector with the degrees of freedom of the model

References

Cinelli, C. and Hazlett, C. (2020), "Making Sense of Sensitivity: Extending Omitted Variable Bias." Journal of the Royal Statistical Society, Series B (Statistical Methodology).

Examples

```
## loads data
data("darfur")

## fits model
model <- lm(peacefactor ~ directlyharmed + age + farmer_dar + herder_dar +
             pastvoted + hhsize_darfur + female + village, data = darfur)

## sensitivity stats for directly harmed
sensitivity_stats(model, treatment = "directlyharmed")

## you can  also pass the numeric values directly
sensitivity_stats(estimate = 0.09731582, se = 0.02325654, dof = 783)
```

Index