Package 'senseweight'

May 9, 2026

Type Package

Title Sensitivity Analysis for Weighted Estimators

Version 0.0.1

Description Provides tools to conduct interpretable sensitivity analyses for weighted estimators, introduced in Huang (2024) <doi:10.1093/jrsssa/qnae012> and Hartman and Huang (2024) <doi:10.1017/pan.2023.12>.
The package allows researchers to generate the set of recommended sensitivity summaries to evaluate the sensitivity in their underlying weighting estimators to omitted moderators or confounders.
The tools can be flexibly applied in causal inference settings (i.e., in external and internal validity contexts)
or survey contexts.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2

Depends R (>= 4.1.0)

LazyData true

Imports dplyr, estimatr, ggplot2, ggrepel, kableExtra, metR, rlang, survey, WeightIt

Suggests knitr, pkgload, rmarkdown

VignetteBuilder knitr

URL https://melodyyhuang.github.io/senseweight/

NeedsCompilation no

Author Melody Huang [aut, cre]

Maintainer Melody Huang <melody.huang@yale.edu>

Repository CRAN

Date/Publication 2025-08-22 17:30:02 UTC

Contents

`benchmark_survey` *Benchmark (for survey weights)*

Description

Returns benchmarking results for survey weighting

Usage

```
benchmark_survey(
  omit,
  formula,
  weights,
  pop_svy = NULL,
  sample_svy,
  Y,
  population_targets = NULL,
  weighting_method = "raking"
)
```

Arguments

`omit`	Variable to benchmark
`formula`	Raking formula
`weights`	A vector, containing the estimated survey weights
`pop_svy`	Survey object, containing the population the survey sample is being re-weighted to
`sample_svy`	Survey object, containing the survey sample being re-weighted
`Y`	outcome of interest
`population_targets`	Population targets for the raking formula (optional, if not provided, will be generated from pop_svy)

weighting_method
: Weighting method (default to raking)

Value

Benchmarking results for a variable (or subset of variables)

Examples

```
data(poll.data)
poll_srs <- survey::svydesign(ids = ~ 1, data = poll.data)
pop_targets = c(1, 0.212, 0.264, 0.236, 0.310,
                0.114, 0.360, 0.528, 0.114,
                0.021, 0.034, 0.805,
                0.266, 0.075, 0.312, 0.349)
names(pop_targets) = c("(Intercept)",
                       "age_buckets36to50",
                       "age_buckets51to64",
                       "age_bucketsOver65",
                       "educHigh School or Less",
                       "educPost-grad",
                       "educSome college",
                       "genderWomen",
                       "raceBlack",
                       "raceHispanic",
                       "raceOther",
                       "raceWhite",
                       "pidIndependent", "pidOther",
                       "pidRepublican", "bornagainYes")
#Set up raking formula:
formula_rake <- ~ age_buckets + educ + gender + race + pid + bornagain

#PERFORM RAKING:
model_rake <- survey::calibrate(
  design = poll_srs,
  formula = formula_rake,
  population = pop_targets,
  calfun = "raking",
  force = TRUE
)


rake_results <- survey::svydesign( ~ 1, data = poll.data, weights = stats::weights(model_rake))
#Estimate from raking results:
weights = stats::weights(rake_results) * nrow(model_rake)

unweighted_estimate = survey::svymean(~ Y, poll_srs, na.rm = TRUE)
weighted_estimate = survey::svymean(~ Y, model_rake, na.rm = TRUE)
benchmark_survey('educ',
                 formula = formula_rake,
                 weights = weights,
                 population_targets = pop_targets,
                 sample_svy = poll_srs,
```

```
                    Y = poll.data$Y)
```

`contour_plot` *Bias Contour Plots*

Description

Generates bias contour plots to aid with sensitivity analysis

Usage

```
contour_plot(
  varW,
  sigma2,
  killer_confounder,
  df_benchmark,
  benchmark = TRUE,
  shade = FALSE,
  shade_var = NULL,
  shade_fill = "#35a4bf",
  shade_alpha = 0.25,
  contour_width = 1,
  binwidth = NULL,
  label_size = 0.25,
  point_size = 1,
  nudge = 0.05,
  axis_text_size = 12,
  axis_title_size = 14,
  axis_line_width = 1,
  print = FALSE
)
```

Arguments

`varW`	Variance of the estimated weights
`sigma2`	Estimated variance of the outcome (i.e., stats::var(Y) for obervational setting; stats::var(tau) for generalization setting)#'
`killer_confounder`	Threshold for bias considered large enough to be a killer confounder. For example, if researchers are concerned about the bias large enough to reduce an estimated treatment effect to zero or change directional sign, set `killer_confounder` equal to the point estimate.
`df_benchmark`	Data frame containing formal benchmarking results. The data.frame must contain the columns `variable` (for the covariate name), `R2_benchmark`, and `rho_benchmark`.

benchmark	Flag for whether or not to display benchmarking results (benchmark = TRUE if we want to add benchmarking results to plot, benchamrk=FALSE otherwise). If set to TRUE, df_benchmark must contain valid benchmarking results.
shade	Flag for whether or not a specific benchmarking covariate (or set of benchmarked covariates) should be shaded a different color (shade = TRUE indicates that we want to highlight specific variables)
shade_var	If shade = TRUE, this contains either a vector containing the variables we want to highlight
shade_fill	Color to fill the highlighted variables. Default is set to "#35a4bf".
shade_alpha	Alpha value for the fill color. Default is set to 0.25.
contour_width	Width of the contour lines. Default is set to 1.
binwidth	If set to a numeric value, the function will generate a contour plot with the specified binwidth. Default is set to NULL.
label_size	Size of the labels. Default is set to 0.25.
point_size	Size of the points. Default is set to 1.
nudge	Nudge value for the labels. Default is set to 0.05.
axis_text_size	Size of the axis text. Default is set to 12.
axis_title_size	Size of the axis title. Default is set to 14.
axis_line_width	Width of the axis lines. Default is set to 1.
print	If set to TRUE, the function will return a list with two elements: plot which contains the generated bias contour plot, and data, which provides the data.frame for generating the contour plot. If set to FALSE, the function will simply generate the bias contour plot. Default is set to FALSE.

Value

A ggplot2 object, containing the bias contour plot

Examples

```
# For the external validity setting:
data(jtpa_women)
site_name <- "NE"
df_site <- jtpa_women[which(jtpa_women$site == site_name), ]
df_else <- jtpa_women[which(jtpa_women$site != site_name), ]

# Estimate unweighted estimator:
model_dim <- estimatr::lm_robust(Y ~ T, data = df_site)
PATE <- coef(lm(Y ~ T, data = df_else))[2]
DiM <- coef(model_dim)[2]
# Generate weights using observed covariates:
df_all <- jtpa_women
df_all$S <- ifelse(jtpa_women$site == "NE", 1, 0)
model_ps <- WeightIt::weightit(
  (1 - S) ~ . - site - T - Y,
```

```
  data = df_all, method = "ebal", estimand = "ATT"
)
weights <- model_ps$weights[df_all$S == 1]
# Estimate IPW model:
model_ipw <- estimatr::lm_robust(Y ~ T, data = df_site, weights = weights)
ipw <- coef(model_ipw)[2]
# Estimate bound for var(tau):
vartau <- var(df_site$Y[df_site$T == 1]) - var(df_site$Y[df_site$T == 0])
RV <- robustness_value(estimate = ipw, b_star = 0, sigma2 = vartau, weights = weights)

# Select weighting variables:
weighting_vars <- names(df_all)[which(!names(df_all) %in% c("site", "S", "Y", "T"))]

# Run benchmarking:
df_benchmark <- run_benchmarking(
  weighting_vars = weighting_vars,
  data = df_all[, -1],
  treatment = "T", outcome = "Y", selection = "S",
  estimate = ipw,
  RV = RV, sigma2 = vartau,
  estimand = "PATE"
)
# Generate bias contour plot:
contour_plot(
var(weights), vartau, ipw, df_benchmark,
benchmark = TRUE, shade = TRUE,
shade_var = c("age", "prevearn"),
label_size = 4
)
```

`create_targets` *Helper function for creating targets from auxiliary information and formula*

Description

Returns weighting targets for survey objects.

Usage

```
create_targets(target_design, target_formula)
```

Arguments

`target_design`	A survey object
`target_formula`	A formula object that contains the variables to weight on

Value

Weighting target for survey objects

Examples

```
data(poll.data)
poll.svy = survey::svydesign(ids = ~ 1,
                     data = poll.data)

#Set up raking formula:
formula_rake = ~ age_buckets + educ + gender + race + educ * pid + bornagain

#Generate targets:
targets_rake = create_targets(poll.svy, formula_rake)
```

`estimate_bias` *Estimate Bias*

Description

Returns the bias based on the different parameters in the bias decomposition

Usage

```
estimate_bias(rho, R2, weights, sigma2)
```

Arguments

`rho`	Correlation between the error in the weights and outcome
`R2`	R2 measure for how much variation in the true weights is explained by the error term, must be bound on the range [0,1)
`weights`	Vector of estimated weights
`sigma2`	Estimated variance of the outcome (i.e., stats::var(Y) for obervational setting; stats::var(tau) for generalization setting)

Value

Estimated bias from omitting a confounder from weights

Examples

```
set.seed(331)
Y = rnorm(1000)
weights = rlogis(1000)
weights = weights/mean(weights)
estimate_bias(rho = 0.5, R2 = 0.5, weights = weights, sigma2 = var(Y))
```

`jtpa_women` *National Jobs Training Partnership Act (JTPA) Experimental Data*

Description

A dataset containing the earnings and other demographic characteristics of individuals identified as women from the JTPA experiment, spanning 16 different experimental sites.

Usage

```
jtpa_women
```

Format

A data frame with 6109 total observations, and 11 columns

site Experimental site that the individual belonged to

Y Earnings, in US dollars

T Treatment assignment status

prevearn Preivous earnings, in US dollars

age Age of individual

married Marital status (1 for married, 0 otherwise)

hrwage Hourly Wage

black Indicator for whether or not the individual is black

hispanic Indicator for whether or not the individual is Hispanic

hsorged Indicator for whether or not the individual received a high school degree, or GED

yrs_educ Number of years of education completed. ...

Source

https://www.upjohn.org/data-tools/employment-research-data-center/national-jtpa-study

poll.data *Synthetic Polling Data*

Description

- pid: party indicator, one of c('Democrat', 'Republican', 'Independent', 'Other')
- educ: education level, one of c('High School or Less', 'Some college', 'College', 'Post-grad')
- age: age of the individual
- age_buckets: grouped age buckets, one of c('18to35', '36to50', '51to64', 'Over65')
- bornagain: indicator for whether the individual identifies as born-again Christian, one of c('Yes', 'No')
- gender: categorical gender, one of c('Men', 'Women')
- race: categorical race, one of c('White', 'Black', 'Hispanic', 'Asian', 'Other')
- vvweight_post: post-election verified voter weight for the individual

Usage

```
data('poll.data')
```

Details

A synthetically generated dataset that simulates polling data for a hypothetical support. The covariates are constructed to match commonly used demographic covariates in practice.

References

"Sensitivity analysis for survey weights," Political Analysis, 32(1) (2024), 1-16. #'

Examples

```
data('poll.data')
```

robustness_value *Robustness Value*

Description

Returns the estimated robustness value, for a specified proportion change

Usage

```
robustness_value(estimate, b_star = 0, sigma2, weights)
```

Arguments

estimate	Weighted estimate
b_star	Threshold that corresponds to a substantively meaningful change to the research conclusion. For example, a value of 0 denotes that the bias from omitting a variable was sufficiently large to change the estimate to zero.
sigma2	Estimated variance of the outcome (i.e., stats::var(Y) for obervational setting; stats::var(tau) for generalization setting)
weights	Vector of estimated weights

Value

Robustness value for a specified proportion change

Examples

```
data(jtpa_women)
site_name <- "NE"
df_site <- jtpa_women[which(jtpa_women$site == site_name), ]
df_else <- jtpa_women[which(jtpa_women$site != site_name), ]

# Estimate unweighted estimator:
model_dim <- estimatr::lm_robust(Y ~ T, data = df_site)
PATE <- coef(lm(Y ~ T, data = df_else))[2]
DiM <- coef(model_dim)[2]
# Generate weights using observed covariates:
df_all <- jtpa_women
df_all$S <- ifelse(jtpa_women$site == "NE", 1, 0)
model_ps <- WeightIt::weightit(
  (1 - S) ~ . - site - T - Y,
  data = df_all, method = "ebal", estimand = "ATT"
)
weights <- model_ps$weights[df_all$S == 1]
# Estimate IPW model:
model_ipw <- estimatr::lm_robust(Y ~ T, data = df_site, weights = weights)
ipw <- coef(model_ipw)[2]
# Estimate bound for var(tau):
vartau <- var(df_site$Y[df_site$T == 1]) - var(df_site$Y[df_site$T == 0])
RV <- robustness_value(estimate = ipw, b_star = 0, sigma2 = vartau, weights = weights)

print(RV)
```

---

run_benchmarking *Run Formal Benchmarking*

Description

Wrapper function to run formal benchmarking on a set of pre-specified covariates. Returns a data.frame containing the benchmarked parameter values, the estimated bias, MRCS, and minimum k_sigma and k_rho values for a killer confounder.

Usage

```
run_benchmarking(
  estimate,
  RV,
  formula = NULL,
  weights = NULL,
  pop_svy = NULL,
  sample_svy = NULL,
  Y = NULL,
  weighting_vars = NULL,
  benchmark_vars = "all",
  data = NULL,
  treatment = NULL,
  outcome = NULL,
  selection = NULL,
  population_targets = NULL,
  weighting_method = "ebal",
  weight_max = Inf,
  sigma2 = NULL,
  estimand = "ATT"
)
```

Arguments

estimate	Weighted estimate
RV	Robustness Value
formula	Raking formula for survey estimand
weights	A vector, containing the estimated survey weights
pop_svy	Survey object, containing the population the survey sample is being re-weighted to
sample_svy	Survey object, containing the survey sample being re-weighted
Y	outcome of interest (used for survey object)
weighting_vars	Vector of variables to use in the weights estimation for ATT or PATE
benchmark_vars	Vector of variables to benchmark parameters for. If benchmark_vars = 'all', benchmarking will be run across all variables included in the weights. If not set to all, benchmarking will be conducted across the covariates included in the vector.
data	A data.frame containing the observed covariates included in the weights; must include variables specified in weighting_vars
treatment	Denotes which variable is the treatment variable
outcome	Denotes which variable is the outcome variable
selection	Denotes which variable is the selection variable
population_targets	Population targets for the raking formula in surveys (optional, if not provided, will be generated from pop_svy)

weighting_method
: Weighting method. Supports weighting methods from the package WeightIt.

weight_max
: Maximum weight to trim at. Default set to Inf.

sigma2
: If estimand = "PATE", sigma2 must specify the bound on treatment effect heterogeneity. For the other two estimands, the function will automatically calculate the sample variance across the control units, or the survey sample.

estimand
: Specifies estimand; possible parameters include "ATT", "PATE", or "Survey"

Value

data.frame containing the benchmarked parameter values, the estimated bias, MRCS, and minimum k_sigma and k_rho values for a killer confounder for the set of pre-specified covariates.

Examples

```
# For the external validity setting:
data(jtpa_women)
site_name <- "NE"
df_site <- jtpa_women[which(jtpa_women$site == site_name), ]
df_else <- jtpa_women[which(jtpa_women$site != site_name), ]

# Estimate unweighted estimator:
model_dim <- estimatr::lm_robust(Y ~ T, data = df_site)
PATE <- coef(lm(Y ~ T, data = df_else))[2]
DiM <- coef(model_dim)[2]
# Generate weights using observed covariates:
df_all <- jtpa_women
df_all$S <- ifelse(jtpa_women$site == "NE", 1, 0)
model_ps <- WeightIt::weightit(
  (1 - S) ~ . - site - T - Y,
  data = df_all, method = "ebal", estimand = "ATT"
)
weights <- model_ps$weights[df_all$S == 1]
# Estimate IPW model:
model_ipw <- estimatr::lm_robust(Y ~ T, data = df_site, weights = weights)
ipw <- coef(model_ipw)[2]
# Estimate bound for var(tau):
vartau <- var(df_site$Y[df_site$T == 1]) - var(df_site$Y[df_site$T == 0])
RV <- robustness_value(estimate = ipw, b_star = 0, sigma2 = vartau, weights = weights)

# Select weighting variables:
weighting_vars <- names(df_all)[which(!names(df_all) %in% c("site", "S", "Y", "T"))]

# Run benchmarking:
df_benchmark <- run_benchmarking(
  weighting_vars = weighting_vars,
  data = df_all[, -1],
  treatment = "T", outcome = "Y", selection = "S",
  estimate = ipw,
  RV = RV, sigma2 = vartau,
  estimand = "PATE"
```

```
)

print(df_benchmark)
```

`summarize_sensitivity` *Sensitivity Summary*

Description

Returns a data.frame or Kable table with summary measures of sensitivity

Usage

```
summarize_sensitivity(
  weights = NULL,
  Y = NULL,
  Z = NULL,
  b_star = 0,
  estimate = NULL,
  SE = NULL,
  unweighted = NULL,
  sigma2 = NULL,
  estimand = "ATT",
  pretty = FALSE,
  svy_srs = NULL,
  svy_wt = NULL,
  sig.fig = 2
)
```

Arguments

`weights`	Vector of estimated weights
`Y`	Outcome of interest
`Z`	Treatment assignment (Not needed for settings when users are analyzing surveys)
`b_star`	Killer confounder threshold. If not specified, will be automatically set to 0.
`estimate`	(Optional) Weighted point estimate. If not specified, function will automatically generate the weighted estimator, given the inputs in `Y` and `weights`
`SE`	(Optional) Standard error associated with the weighted point estimate
`unweighted`	(Optional) Unweighted point estimate.
`sigma2`	(Optional) Variance of outcomes or individual-level treatment effect in PATE case. In the case of a PATE estimator, if not specified, function will automatically estimate an upper bound for the variance of the individual-level treatment effect.

estimand	Specifies estimand; possible parameters include "ATT", "PATE", or "Survey"
pretty	If set to TRUE, will return a Kable table. If set to FALSE, will return a data.frame.
svy_srs	Unweighted svymean object
svy_wt	Weighted svymean object
sig.fig	Significant figures to round the output to (default set to 2)

Value

Sensitivity summary

Examples

```
data(jtpa_women)
site_name <- "NE"
df_site <- jtpa_women[which(jtpa_women$site == site_name), ]
df_else <- jtpa_women[which(jtpa_women$site != site_name), ]

# Estimate unweighted estimator:
model_dim <- estimatr::lm_robust(Y ~ T, data = df_site)
PATE <- coef(lm(Y ~ T, data = df_else))[2]
DiM <- coef(model_dim)[2]
# Generate weights using observed covariates:
df_all <- jtpa_women
df_all$S <- ifelse(jtpa_women$site == "NE", 1, 0)
model_ps <- WeightIt::weightit(
  (1 - S) ~ . - site - T - Y,
  data = df_all, method = "ebal", estimand = "ATT"
)
weights <- model_ps$weights[df_all$S == 1]
# Estimate IPW model:
model_ipw <- estimatr::lm_robust(Y ~ T, data = df_site, weights = weights)
ipw <- coef(model_ipw)[2]
# Estimate bound for var(tau):
vartau <- var(df_site$Y[df_site$T == 1]) - var(df_site$Y[df_site$T == 0])
summarize_sensitivity(weights = weights,
Y = df_site$Y,
Z = df_site$T,
sigma2 = vartau,
 estimand = "PATE")
```

---

summarize_sensitivity_survey

Sensitivity Summary (for survey weights)

Description

Returns a data.frame or Kable table with summary measures of sensitivity; helper function for main summary function, and allows users to directly input a survey object and a design object

Usage

```
summarize_sensitivity_survey(svy_srs, svy_wt, weights, varY, b_star = 0)
```

Arguments

svy_srs	Survey object, containing the unweighted survey
svy_wt	Survey object, containing the weighted survey
weights	A vector, containing the estimated survey weights
varY	variance of the outcome
b_star	Killer confounder threshold, default set to be zero

Value

data.frame with summary measures of sensitivity

Examples

```
data(poll.data)
poll_srs <- survey::svydesign(ids = ~ 1, data = poll.data)
pop_targets = c(1, 0.212, 0.264, 0.236, 0.310,
                0.114, 0.360, 0.528, 0.114,
                0.021, 0.034, 0.805,
                0.266, 0.075, 0.312, 0.349)
names(pop_targets) = c("(Intercept)",
                       "age_buckets36to50",
                       "age_buckets51to64",
                       "age_bucketsOver65",
                       "educHigh School or Less",
                       "educPost-grad",
                       "educSome college",
                       "genderWomen",
                       "raceBlack",
                       "raceHispanic",
                       "raceOther",
                       "raceWhite",
                       "pidIndependent", "pidOther",
                       "pidRepublican", "bornagainYes")
#Set up raking formula:
formula_rake <- ~ age_buckets + educ + gender + race + pid + bornagain

#PERFORM RAKING:
model_rake <- survey::calibrate(
  design = poll_srs,
  formula = formula_rake,
  population = pop_targets,
  calfun = "raking",
  force = TRUE
)
```

```
rake_results <- survey::svydesign( ~ 1, data = poll.data, weights = stats::weights(model_rake))
#Estimate from raking results:
weights = stats::weights(rake_results) * nrow(model_rake)

unweighted_estimate = survey::svymean(~ Y, poll_srs, na.rm = TRUE)
weighted_estimate = survey::svymean(~ Y, model_rake, na.rm = TRUE)
summarize_sensitivity(estimand = 'Survey',
Y = poll.data$Y,
weights = weights,
svy_srs = unweighted_estimate,
svy_wt = weighted_estimate,
b_star = 0.5)
```

Index