Package: vbm (via r-universe)

July 1, 2026

Type Package

Title Variance-Based Sensitivity Analysis for Weighting Estimators

Version 0.1.0

Maintainer Jiayao Gan <`u3612852@connect.hku.hk`>

Description Provides methods for variance-based sensitivity analysis and weighting estimators in observational studies based on methodology by Huang & Pimentel (2025) <doi:10.1093/biomet/asae040>. Includes bootstrap inference, bias bounds estimation, and visualization tools for sensitivity parameters.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Imports parallel, magrittr, dplyr, WeightIt, estimatr, ggplot2, scales

URL `https://github.com/Staniks0/vbm`

BugReports `https://github.com/Staniks0/vbm/issues`

RoxygenNote 7.3.3

Suggests jointVIP, knitr, rmarkdown, pkgdown, cobalt, osqp

VignetteBuilder knitr

NeedsCompilation no

Author Jiayao Gan [aut, cre], Melody Huang [aut], Samuel D. Pimentel [aut], Andy A. Shen [aut], National Science Foundation [fnd] (Grant #2142146)

Repository https://cran.r-universe.dev

Date/Publication 2026-06-30 20:30:02 UTC

RemoteUrl https://github.com/cran/vbm

RemoteRef HEAD

RemoteSha 889ec8d00600b8e75dd67a42a207dae20f5a4433

Contents

`benchmark_lambda` *Benchmark Lambda Values for Marginal Sensitivity Model*

Description

This function computes a benchmarked lambda value for the Marginal Sensitivity Model (MSM) by omitting a specified covariate from the weighting model. The resulting lambda represents the worst-case odds ratio bound that would be required to account for the confounding due to omitting that covariate. This allows researchers to calibrate sensitivity parameters against observed covariates (Hsu & Small, 2013; Soriano et al., 2023).

Usage

```
benchmark_lambda(omit, weightlist, data, treatment_name = "treatment")
```

Arguments

`omit`	A character string specifying the name of the covariate to omit from the weighting model when computing benchmark weights.
`weightlist`	A pre-fitted weightit object from the WeightIt package. Must contain the full weighting specification including all covariates, method, and any additional parameters (e.g., tols, maxit).
`data`	A data frame containing the covariates and treatment indicator.
`treatment_name`	Character string naming the treatment variable in data. Default is "treatment".

Value

A data frame with one row containing:

`variable`	The name of the omitted variable
`lambda`	The benchmarked lambda value (rounded to 1 decimal place)

Note

This function assumes that the original weights were estimated using the full set of covariates including the omitted variable.

The benchmarked lambda represents the worst-case individual-level deviation in the odds of treatment that would result from omitting the given covariate. Larger lambda values indicate that the covariate is more important for addressing confounding.

References

Hsu, J. Y., & Small, D. S. (2013). Calibrating sensitivity analyses to observed covariates in observational studies. Biometrics, 69(4), 803-811.

Soriano, D., Ben-Michael, E., Bickel, P. J., Feller, A., & Pimentel, S. D. (2023). Interpretable sensitivity analysis for balancing weights. Journal of the Royal Statistical Society Series A: Statistics in Society, 186(4), 707-721.

Examples

```
library(WeightIt)
data("nhanes-clean")
 weightlist <- WeightIt::weightit(treatment ~ gender + age + income + income.missing
                       + education + smoking.now + smoking.ever + race,
                       data = nhanes.clean,
                       method = "ebal",
                       estimand = "ATT")
# Benchmark omitting "education"
benchmark_result <- benchmark_lambda(
  omit = "education",
  weightlist,
  nhanes.clean
)

print(benchmark_result)

# Interpret the result
# If critical lambda* from sensitivity analysis is 5, and benchmarked lambda
# for "education" is 4.4, then an unmeasured confounder would need to be stronger
# than "education" to explain away the effect.

# glm call for weighting, must specify formula, data in glm
glm_call <- quote(glm(
              formula=treatment ~ age + education + income ,
              data = nhanes.clean,family = binomial()
              ))
glm_result <- benchmark_lambda(
  omit = "education",
  glm_call,
  nhanes.clean
)
print(glm_result)
```

`benchmark_R2` *Benchmark R-squared for Leave-One-Out Covariate Omission*

Description

Computes benchmark R-squared, bias, and correlation values for a given omitted covariate by refitting the weighting model without that covariate. This function is used to assess how sensitive results are to unobserved confounders similar to observed covariates.

Usage

```
benchmark_R2(
  omit,
  weightlist,
  data,
  Y = "Y",
  treatment = "treatment",
  estimate
)
```

Arguments

`omit`	A character string specifying the name of the covariate to omit from the weighting model.
`weightlist`	A pre-fitted weightit object from the WeightIt package. Must contain the full weighting specification including all covariates, method, and any additional parameters (e.g., tols, maxit).
`data`	Data frame containing the original data used to fit W. Must include all variables from the weighting formula plus the outcome and treatment variables.
`Y`	Character string specifying the name of the outcome variable in data. Default is "Y".
`treatment`	Character string specifying the name of the treatment variable in data. Default is "treatment".
`estimate`	Numeric value of the treatment effect estimate from the full weighting model (e.g., IPW estimate). Used for calculating additional metrics like MRCS if needed.

Value

A data frame with one row containing the following columns:

`variable`	The name of the omitted covariate
`bias`	The estimated bias from omitting the covariate
`R2_benchmark`	The R-squared value representing the proportion of variance in weights explained by the omitted covariate
`rho_benchmark`	The correlation between the weight difference (full - benchmark) and the outcome among control units

References

Huang, M., & Pimentel, S. D. (2025). Variance-based sensitivity analysis for weighting estimators results in more informative bounds. *Biometrika*, 112(1).

Examples

```
library(WeightIt)
weightlist <- weightit(treatment ~ age + education + income,
                  data = nhanes.clean,
                  method = "ebal",
                  estimand = "ATT",
                  tols = 0.01)
# First get the IPW estimate
model_ipw <- estimatr::lm_robust(Y ~ treatment, data = nhanes.clean, weights = weightlist$weights)
ipw_estimate <- coef(model_ipw)[2]

# Benchmark R-squared for omitting "age"
result <- benchmark_R2(
  omit = "age",
  weightlist = weightlist,
  data = nhanes.clean,
  estimate = ipw_estimate
)

print(result)

# glm call
glm_call <- quote(glm(formula = treatment ~ age + education + gender + income,
    data = nhanes.clean,family = binomial()))
result <- benchmark_R2(
  omit = "age",
  weightlist = glm_call,
  data = nhanes.clean,
  estimate = ipw_estimate
)
print(result)
```

`estimate_confidence_intervals` *Compute Confidence Intervals for Sensitivity Analysis*

Description

This function performs a percentile bootstrap procedure to construct confidence intervals for the Average Treatment Effect on the Treated (ATT) under either the Marginal Sensitivity Model (MSM) or the Variance-Based Sensitivity Model (VBM). The intervals account for both sampling uncertainty and unmeasured confounding, following the methods of Zhao et al. (2019) and Soriano et al. (2023).

Usage

```
estimate_confidence_intervals(
  alpha = 0.05,
  weightlist,
  df,
  lambda_seq = seq(1, 8, by = 0.25),
  R2_seq = seq(0, 0.9, by = 0.01),
  cor_eps_seq = NULL,
  treatment_variable = "treatment",
  approach,
  covariates = NULL,
  n_bootstrap = 1000,
  n_cores = 4,
  seed = 331
)
```

Arguments

`alpha`	Significance level for confidence intervals (default = 0.05 for 95% CI)
`weightlist`	A pre-fitted weightit object from the WeightIt package. Must contain the estimated weights and the original call specification.
`df`	Data frame containing the original data used to fit W. Must include all variables from the weighting formula plus the outcome.
`lambda_seq`	Numeric vector of lambda values for MSM approach. Default = seq(1, 8, by = 0.25). Lambda bounds the odds ratio deviation from ignorability.
`R2_seq`	Numeric vector of R-squared values for VBM approach. Default = seq(0, 0.9, by = 0.01). R-squared represents proportion of variance in weights from unobserved confounders.
`cor_eps_seq`	Numeric vector of correlation values for VBM with correlation approach. Must be same length as R2_seq. If NULL, uses worst-case bounds.
`treatment_variable`	Character string naming the treatment variable in df. Default = "treatment".
`approach`	Character string specifying the sensitivity approach. Must be one of: "msm" (Marginal Sensitivity Model), "vbm" (Variance-Based Model), or "vbm_w_cor" (VBM with user-specified correlation).
`covariates`	Optional character vector of variable names to add to the output data frame. If provided, each variable name will be repeated for all rows.
`n_bootstrap`	Integer number of bootstrap iterations. Default = 1000.
`n_cores`	Integer number of CPU cores for parallel computation. Default = 4.
`seed`	Integer seed for reproducibility. Default = 331.

Value

A data frame containing confidence intervals for the ATT across sensitivity parameter values:

- For `approach = "msm"`: Columns `lambda`, `ci_low`, `ci_high`

- For `approach = "vbm"`: Columns `R2`, `ci_low`, `ci_high` (and `rho` if `cor_eps_seq` provided)
- If `covariates` is provided: an additional column `variable`

Note

This function uses parallel processing via `parallel::mclapply()` with 4 cores. For Windows systems, you may need to adjust the parallel backend.

The function assumes the outcome variable is named `"Y"` in the input data frame.

References

Zhao, Q., Small, D. S., & Bhattacharya, B. B. (2019). Sensitivity analysis for inverse probability weighting estimators via the percentile bootstrap. Journal of the Royal Statistical Society: Series B, 81(4), 735-761.

Soriano, D., Ben-Michael, E., Bickel, P. J., Feller, A., & Pimentel, S. D. (2023). Interpretable sensitivity analysis for balancing weights. Journal of the Royal Statistical Society Series A: Statistics in Society, 186(4), 707-721.

Huang, M., & Pimentel, S. D. (2025). Variance-based sensitivity analysis for weighting estimators results in more informative bounds. Biometrika.

Examples

```
# MSM confidence intervals
library(WeightIt)

# First, fit the weighting model
weightlist_msm <- weightit(treatment ~ age + education + income,
                  data = nhanes.clean,
                  method = "ebal",
                  estimand = "ATT")

# Run MSM confidence intervals
msm_results <- estimate_confidence_intervals(
  alpha = 0.05,
  weightlist = weightlist_msm,
  df = nhanes.clean,
  lambda_seq = seq(1, 5, by = 0.5),
  treatment_variable = "treatment",
  approach = "msm",
  n_bootstrap = 1000,
  n_cores = 1,
  seed = 123
)

print(msm_results)


# VBM confidence intervals
weightlist_vbm <- weightit(treatment ~ age + education + income,
                  data = nhanes.clean,
```

```
                  method = "optweight",
                  estimand = "ATT",
                  tols = 0.01)

vbm_results <- estimate_confidence_intervals(
  alpha = 0.05,
  weightlist = weightlist_vbm,
  df = nhanes.clean,
  R2_seq = seq(0, 0.9, by = 0.1),
  treatment_variable = "treatment",
  approach = "vbm",
  n_bootstrap = 1000,
  n_cores = 1,
  seed = 123
)

print(vbm_results)


# VBM with user-specified correlation and custom numbers of bootstrap iterations and cores
weightlist_vbm_cor <- weightit(treatment ~ age + education + income,
                      data = nhanes.clean,
                      method = "optweight",
                      estimand = "ATT")

vbm_results_cor <- estimate_confidence_intervals(
  alpha = 0.05,
  weightlist = weightlist_vbm_cor,
  df = nhanes.clean,
  R2_seq = seq(0, 0.9, by = 0.1),
  cor_eps_seq = rep(0.9, 10),
  treatment_variable = "treatment",
  approach = "vbm",
  n_bootstrap = 500,
  n_cores = 1,
  seed = 123
)

print(vbm_results_cor)

# glm call
glm_call <-quote(glm(treatment ~ age + education + income,
                data = nhanes.clean,
                family = binomial()))
glm_results <- estimate_confidence_intervals(
  alpha = 0.05,
  weightlist = glm_call,
  df = nhanes.clean,
  lambda_seq = seq(1, 5, by = 0.5),
  treatment_variable = "treatment",
  approach = "msm",
  n_bootstrap = 1000,
  n_cores = 1,
```

```
  seed = 123
)
print(glm_results)
```

`estimate_point_estimate_bounds` *Compute Point Estimate Bounds for Sensitivity Analysis*

Description

This function computes the range of possible point estimates (i.e., bounds) for the Average Treatment Effect on the Treated (ATT) under either the Marginal Sensitivity Model (MSM) or the Variance-Based Sensitivity Model (VBM). These bounds represent the worst-case treatment effects possible given a specified level of unmeasured confounding, without accounting for sampling uncertainty.

Usage

```
estimate_point_estimate_bounds(
  weights,
  Y,
  treatment,
  approach = "vbm",
  lambda_range = seq(1, 8, by = 0.25),
  R2_range = seq(0, 0.9, by = 0.01),
  variables = NULL,
  data
)
```

Arguments

`weights`	A numeric vector of balancing weights (estimated from observed data). Typically obtained from `WeightIt::weightit()` or similar.
`Y`	A numeric vector of outcomes.
`treatment`	A numeric or logical vector indicating treatment assignment (1 = treated, 0 = control).
`approach`	A character string specifying the sensitivity model: `"msm"` for Marginal Sensitivity Model or `"vbm"` for Variance-Based Model. Default is `"vbm"`.
`lambda_range`	A numeric vector of lambda values (>= 1) for the MSM. Default is `seq(1, 8, by = 0.25)`.
`R2_range`	A numeric vector of R-squared values (0 to 1) for the VBM. Default is `seq(0, 0.9, by = 0.01)`.
`variables`	Optional character vector of variable names to add to the output data frame. If provided, each variable name will be repeated for all rows.
`data`	Optional data frame containing the variables. Currently not used but may be included for compatibility with other functions.

Value

A data frame containing the point estimate bounds:

- For `approach = "msm"`: Columns `lambda`, `ATT_low`, `ATT_high`
- For `approach = "vbm"`: Columns `R2`, `ATT_low`, `ATT_high`
- If `variables` is provided: an additional column `variable`

Note

These point estimate bounds do NOT account for sampling uncertainty. For confidence intervals that incorporate sampling variability, use a bootstrap procedure (e.g. `estimate_confidence_intervals`).

The bounds represent the range of possible point estimates under the specified confounding level. As lambda increases (MSM) or R2 increases (VBM), the bounds widen.

References

Soriano, D., Ben-Michael, E., Bickel, P. J., Feller, A., & Pimentel, S. D. (2023). Interpretable sensitivity analysis for balancing weights. Journal of the Royal Statistical Society Series A: Statistics in Society, 186(4), 707-721.

Zhao, Q., Small, D. S., & Bhattacharya, B. B. (2019). Sensitivity analysis for inverse probability weighting estimators via the percentile bootstrap. Journal of the Royal Statistical Society: Series B, 81(4), 735-761.

Huang, M., & Pimentel, S. D. (2025). Variance-based sensitivity analysis for weighting estimators results in more informative bounds. Biometrika.

Examples

```
# MSM approach
library(WeightIt)
weightlist <- weightit(treatment ~ age + education, data = nhanes.clean,
              method = "ps", estimand = "ATT")
weights <- weightlist$weights
bounds_msm <- estimate_point_estimate_bounds(
  weights = weights,
  Y = nhanes.clean$Y,
  treatment = nhanes.clean$treatment,
  approach = "msm",
  lambda_range = seq(1, 5, by = 0.5)
)
print(bounds_msm)

# VBM approach
bounds_vbm <- estimate_point_estimate_bounds(
  weights = weights,
  Y = nhanes.clean$Y,
  treatment = nhanes.clean$treatment,
  approach = "vbm",
  R2_range = seq(0, 0.9, by = 0.1)
)
```

```
print(bounds_vbm)

# With variable names for plotting
bounds_with_names <- estimate_point_estimate_bounds(
  weights = weights,
  Y = nhanes.clean$Y,
  treatment = nhanes.clean$treatment,
  approach = "msm",
  lambda_range = c(2,3),
  variables = c("age","education")
)
print(bounds_with_names)
```

`ncds` *National Child Development Survey (NCDS) Data*

Description

The National Child Development Survey (NCDS) is a longitudinal study of individuals born in the United Kingdom during the week of 3-9 March 1958. The original study collected information on educational attainment, familial backgrounds, and socioeconomic and health well-being for 17,415 individuals.

Usage

```
ncds
```

Format

A data frame with 3,642 rows and 14 columns.

Details

Following Battistin and Sianesi (2011), the data were pre-processed to obtain a subset of 3,642 males who were employed in 1991 and had complete educational attainment and wage information. Missing covariates were imputed a single time using Multiple Imputation by Chained Equations (MICE; Buuren and Groothuis-Oudshoorn, 2010).

Outcome Variable:

- `wage`: Log gross hourly wage in British Pounds.

Treatment Variables:

- `Dany` (binary): Indicator of any academic qualification. Levels: 1 = qualification (n = 2,399), 0 = no qualification (n = 1,243).

Covariates (Pre-treatment Confounders):

- `white`: Indicator of self-identified white race.

- `maemp`: Mother's employment status in 1974.
- `scht`: School type attended at age 16.
- `qmab`, `qmab2`: Mathematics test scores at ages 7 and 11.
- `qvab`, `qvab2`: Reading test scores at ages 7 and 11.
- `paed_u`, `maed_u`: Father and mother's years of education.
- `sib_u`: Number of siblings.
- `agepa`, `agema`: age of father and mother in 1974.

Use `str(NCDS)` to inspect the full structure of the dataset.

Source

Battistin, E., & Sianesi, B. (2011). Misclassified Treatment Status and Treatment Effects: An Application to Returns to Education in the United Kingdom. *Review of Economics and Statistics*, 93(2), 495-509.

References

https://cls.ucl.ac.uk/cls-studies/1958-national-child-development-study/

Battistin E, Sianesi B. (2011). Misclassified Treatment Status and Treatment Effects: an Application to Returns to Education in the United Kindom. Review of Economics and Statistics, 93(2), 495-509.

Battistin E, Sianesi B. (2012). Replication data for: Misclassified Treatment Status and Treatment Effects: an Application to Returns to Education in the United Kindom. URL https://doi.org.10.7910/DVN/EPCYUL.

`nhanes.clean` *NHANES 2013-2014 Mercury and Fish Consumption Data*

Description

A cleaned subset of data from the 2013-2014 National Health and Nutrition Examination Survey (NHANES), used for sensitivity analysis in causal inference studies of fish consumption on blood mercury levels.

Usage

```
nhanes.clean
```

Format

A data frame with 1,107 rows and 9 columns:

Y Total blood mercury (in log2 scale, micrograms per liter). A one-unit difference implies a twofold difference in total blood mercury.

treatment Binary indicator of fish/shellfish consumption in the preceding month. 1 = more than 12 servings, 0 = 12 or fewer servings.

gender Participant gender.

age Participant age in years.

income Household income level.

income.missing Whether income data is missing.

race Participant race/ethnicity.

education Educational attainment level.

smoking.now Current smoking status.

smoking.ever Lifetime smoking history (ever smoked or not).

Details

The outcome variable is total blood mercury measured in micrograms per liter, transformed to log2 scale. An estimated treated-control outcome difference of one implies that a treated person's total blood mercury is twice that of a control.

The treatment variable indicates whether an individual consumed more than 12 servings of fish or shellfish in the month preceding the survey.

The dataset contains:

- 234 treated units (consumed larger than 12 servings of fish/shellfish)
- 873 control units (consumed no larger than 12 servings of fish/shellfish)

Demographic variables (gender, age, income, race, education, and smoking history) are included as covariates to adjust for non-random treatment assignment. These are commonly used to estimate propensity score weights via logistic regression for causal analyses of fish consumption on blood mercury levels.

Source

National Health and Nutrition Examination Survey (NHANES) 2013-2014. `https://www.cdc.gov/nchs/nhanes/`

References

Original study describing this data preprocessing approach. (Add citation if available.)

Examples

```
data("nhanes.clean")
head(nhanes.clean)
```

`plot_sensitivity_analysis`

Plot Sensitivity Analysis Results

Description

Creates a comprehensive visualization of sensitivity analysis results showing how treatment effect estimates and confidence intervals vary across different levels of unmeasured confounding. The function generates a publication-ready ggplot2 object with customizable options for benchmarks, axis breaks, and titles.

Usage

```
plot_sensitivity_analysis(
  results,
  parameter_level = NULL,
  ci = TRUE,
  benchmarking = FALSE,
  benchmark_variable = NULL,
  variable_name = NULL,
  header = NULL,
  x_axis_breaks = "default",
  x_break_value = NULL
)
```

Arguments

`results` A list object returned by `run_sensitivity_analysis` containing the following components:

- `point_bounds`: Data frame with columns `ATT_low`, `ATT_high`, and parameter values (`R2` or `lambda`)
- `confidence_intervals`: Data frame with columns `ci_low`, `ci_high`, and parameter values
- `Rstar` or `lambda_star`: Numeric critical threshold value
- `ipw_estimate`: Numeric IPW estimate at parameter = 0
- `approach`: Character string specifying the sensitivity analysis approach. Must be either `"vbm"`, `"vbm_w_cor"` (uses R^2 parameter), or any other approach name (uses λ parameter). Determines x-axis labeling and critical value extraction.
- `benchmark_parameters`: Data frame with benchmark variables (required if `benchmarking = TRUE`)

`parameter_level`

Numeric vector of parameter values to display on the x-axis. Values are rounded to 5 decimal places for matching. If `NULL` (default), all available parameter values from the analysis are shown. Use this to focus on a specific range of interest.

`ci`	Logical flag indicating whether there is confidence interval evaluation in `results`
`benchmarking`	Logical flag indicating whether to add benchmark reference lines from observed confounders. When `TRUE`, benchmark reference lines are plotted. Default is `FALSE`.
`benchmark_variable`	Character vector specifying which benchmark variables to display as reference lines. If `NULL` (default), all benchmark variables in `results$benchmark_parameters` are shown.
`variable_name`	Character vector for renaming benchmark variables in the plot. Must have the same length as the selected `benchmark_variable` after filtering if `benchmark_variable` is not `NULL` (default). Otherwise should be as long as the benchmark variables in `results$benchmark_parameters`. Use this to display more descriptive labels. Default is `NULL`.
`header`	Character string for the plot title. If `NULL` (default), no title is added to the plot.
`x_axis_breaks`	Character string controlling x-axis break behavior. Options: • `"default"`: Let ggplot automatically determine breaks • `"pretty"`: Use `scales::pretty_breaks()` for nice round numbers • `"custom"`: Use user-specified interval (requires `x_break_value`, otherwise falls back to `"default"`) • `"none"`: Display no axis breaks Default is `"default"`.
`x_break_value`	Numeric value specifying the interval between custom breaks. Only used when `x_axis_breaks = "custom"`. The breaks will be generated as `seq(0, max(df_bounds$param), by = x_break_value)`. Default is `NULL`.

Details

The plot includes:

1. Point estimate bounds as vertical line segments
2. 95 percent confidence intervals as error bars
3. A dashed vertical line at the critical threshold (R^* or λ^*)
4. An IPW point estimate at the reference parameter value of 0
5. Optional benchmark reference lines from observed confounders
6. A horizontal reference line at $y = 0$ (no treatment effect)

Colors indicate whether estimates fall below (steelblue) or above (slategray) the critical threshold. The x-axis represents the confounding parameter (R^2 for VBM approaches, λ for other approaches).

The function performs several data processing steps before plotting:

1. Extracts point bounds and confidence intervals from the `results` list.
2. Determines approach-specific parameters (R^2 for VBM, λ for others).
3. Optionally filters data to specified `parameter_level` values.
4. Color-codes estimates based on position relative to critical threshold.

5. Optionally filters and renames benchmark variables.
6. Constructs the ggplot with appropriate scales and themes.
7. Parameter values are rounded to 5 decimal places when matching to `parameter_level` to avoid floating-point precision issues. Warnings are issued when requested parameter levels or benchmark variables are not found.

Value

A `ggplot2` object that can be further customized using standard ggplot2 syntax (e.g., adding + `theme(...)` or + `labs(...)`). The plot displays:

- X-axis: Confounding parameter (R^2 or λ)
- Y-axis: Estimated Average Treatment Effect (ATT)
- Point estimates and confidence intervals colored by critical threshold
- Optional benchmark reference lines with labels

Note

The IPW estimate is always plotted at `param = 0` regardless of whether 0 is included in `parameter_level`. This represents the unadjusted treatment effect estimate.

See Also

1. `run_sensitivity_analysis` for generating the input list `results`
2. `benchmark_lambda` for generating benchmark parameters
3. `ggplot` for additional plot customization options

Examples

```
# Basic usage with default settings
weightlist <- WeightIt::weightit(treatment ~ gender + age + income + income.missing
                      + education + smoking.now + smoking.ever + race,
                      data = nhanes.clean,
                      method = "ps",
                      estimand = "ATT")
results <- run_sensitivity_analysis(weightlist=weightlist,
                                    Y="Y",
                                    treatment="treatment",
                                    data=nhanes.clean,
                                    approach="vbm",
                                    R2_seq = seq(0, 0.8, by = 0.01),
                                    alpha = 0.05,
                                    benchmarking = TRUE,
                                    n_bootstrap = 1000,
                                    n_cores = 1,
                                    seed = 331)
# Add benchmark references with custom labels
plot_sensitivity_analysis(results=results, benchmarking=TRUE,
                           benchmark_variable = c(
```

```
                                 "age","education","smoking.now","smoking.ever","race"),
                        variable_name=c("age","education","smoking","smoking history","race"))

# Focus on specific parameter range with custom title
plot_sensitivity_analysis(results=results, parameter_level=seq(0, 0.6, by = 0.1),,
                          header = "Sensitivity Analysis: R² Range 0-0.6",
                          x_axis_breaks = "pretty")

# Custom x-axis breaks
plot_sensitivity_analysis(results=results,
                          x_axis_breaks = "custom",
                          x_break_value = 0.02)

# Minimal plot without benchmarks or title
plot_sensitivity_analysis(results, x_axis_breaks = "none")
```

`run_sensitivity_analysis` *Run Sensitivity Analysis for Unmeasured Confounding*

Description

Performs sensitivity analysis to assess the robustness of average treatment effect estimates (ATT) to unmeasured confounding using various approaches including Huang & Pimentel's variance-based sensitivity model (vbm), variance-based sensitivity model using a relaxed correlation bound (vbm_w_cor) and Zhao's marginal sensitivity model with bootstrap confidence interval (msm). The function handles weighting, bootstrap confidence intervals, and optional benchmarking against observed confounders.

Usage

```
run_sensitivity_analysis(
  weightlist = NULL,
  weights = NULL,
  Y,
  treatment,
  data,
  approach,
  lambda_seq = seq(1, 8, by = 0.25),
  R2_seq = seq(0, 0.9, by = 0.01),
  cor_eps_seq = NULL,
  alpha = 0.05,
  benchmarking = FALSE,
  n_bootstrap = 1000,
  n_cores = 4,
  seed = 123,
```

```
  verbose = TRUE
)
```

Arguments

`weightlist`	A WeightIt object from the `WeightIt` package or a glm call. The WeightIt object should contain the estimated weights and the original call, while the glm call should contain formula and data argument.
`weights`	A vector of weights fitted on data. Should have the same length as data. Confidence interval and benchmarking will be automatically turned off. If neither of weightlist or weights are provided, execution will halt and throw an error.
`Y`	A character string specifying the name of the outcome variable in `data`.
`treatment`	A character string specifying the name of the treatment variable in `data`.
`data`	A data frame containing the original data used to fit `weightlist`. Must include all variables used in the weighting formula plus the outcome variable.
`approach`	A character string specifying the sensitivity analysis approach. Must be one of: `"msm"` for Marginal Sensitivity Model, `"vbm"` for Variance-Based Model, or `"vbm_w_cor"` for Variance-Based Model with user-specified correlation.
`lambda_seq`	A numeric vector of lambda values for MSM approach. Default is `seq(1, 8, by = 0.25)`. Lambda represents the maximum odds ratio deviation from ignorability.
`R2_seq`	A numeric vector of R-squared values for VBM approach. Default is `seq(0, 0.9, by = 0.01)`. R-squared represents the proportion of variance in weights explained by unobserved confounders.
`cor_eps_seq`	A numeric vector of correlation values for `"vbm_w_cor"` approach. Must be the same length as `R2_seq`. If `NULL` and approach is `"vbm_w_cor"`, the function will use worst-case correlation bounds. Default is `NULL`.
`alpha`	A numeric value specifying the significance level for confidence intervals. Default is `0.05` (95 percent confidence intervals).
`benchmarking`	A logical value indicating whether to perform benchmarking using leave-one-out covariate omission. Default is `FALSE`.
`n_bootstrap`	An integer specifying the number of bootstrap iterations for confidence interval estimation. Default is `1000`.
`n_cores`	An integer specifying the number of CPU cores to use for parallel bootstrap computation. Default is `4`.
`seed`	An integer seed for random number generation to ensure reproducibility. Default is `123`.
`verbose`	A boolean specifying whether give detailed massage at each stage. Default is `TRUE`.

Details

The function implements three sensitivity analysis approaches:

- **VBM (Huang & Pimentel's variance-based sensitivity model):** Uses the R2 parameter to quantify the proportion of residual variance in the outcome explained by unmeasured confounders.
- **MSM (Zhao's marginal sensitivity model):** Uses the lambda parameter representing the ratio of largest possible error on individual weights that can arise from omitting a confounder.
- **VBM with Correlation:** Extends VBM by incorporating correlation between the error terms of the treatment imbalance and outcome.

Value

A list object of class `sensitivity_analysis` containing:

- `difference_in_means`: Unadjusted difference-in-means estimate
- `ipw_estimate`: Inverse probability weighted treatment effect estimate
- `difference_in_means_se`: Standard error of unadjusted estimate
- `ipw_se`: Standard error of IPW estimate
- `approach`: Sensitivity analysis method used
- `point_bounds`: Data frame with parameter values and corresponding bounds for the point estimate of ATT
- `confidence_intervals`: Data frame with parameter values and confidence intervals
- `Rstar`: (if `approach = "vbm"` or `approach = "vbm_w_cor"`) Critical threshold value for R-squared
- `lambda_star`: (if `approach = "msm"`) Critical threshold value for lambda
- `benchmark_parameters`: (if `benchmarking = TRUE`) Benchmark values. For `approach = "msm"`: A data frame with variable names and benchmarked lambdas. For `approach = "vbm"` or `approach = "vbm_w_cor"`: A data frame with variable names, benchmarked R-squared values, correlations, and the maximum bias of point estimates
- `benchmark_point_bounds`: (if `benchmarking = TRUE`) Point bounds at benchmark values
- `benchmark_confidence_intervals`: (if `benchmarking = TRUE`) Confidence intervals at benchmark values

Note

- The dataset must have no missing values in treatment, outcome, or covariates.
- Either `weightlist` or `weights` should be specified. Otherwise, analysis will halt and give a warning message.
- Estimand must be ATT if `weightlist` is a WeightIt object. Otherwise, analysis will halt and give a warning message.
- `formula` and `data` should be specified if if `weightlist` is a glm call. Otherwise, analysis will halt and give a warning message.
- Treatment variable should be binary (0/1).
- For `approach = "vbm_w_cor"`, `cor_eps_seq` must be provided and have the same length as `R2_seq`. Otherwise, a warning will be issued and execution will halt.
- Bootstrap procedures can be computationally intensive for large datasets or many parameter values.

References

- Huang, M., & Pimentel, S. D. (2025). Variance-based sensitivity analysis for weighting estimators results in more informative bounds. *Biometrika*, 112(1).
- Zhao, Q., Small, D. S., & Bhattacharya, B. B. (2019). Sensitivity analysis for inverse probability weighting estimators via the percentile bootstrap. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 81(4), 735-761.

See Also

`plot_sensitivity_analysis` for visualizing results, `estimate_point_estimate_bounds` for bound calculations, `estimate_confidence_intervals` for bootstrap CI computation, `benchmark_lambda` for MSM benchmark calculations

Examples

```
data("nhanes-clean")
weightlist <- WeightIt::weightit(treatment ~ gender + age + income + income.missing
                      + education + smoking.now + smoking.ever + race,
                      data = nhanes.clean,
                      method = "ps",
                      estimand = "ATT")
# Basic MSM sensitivity analysis
results <- run_sensitivity_analysis(weightlist=weightlist,
                                     Y="Y",
                                     treatment="treatment",
                                     data=nhanes.clean,
                                     approach="msm",
                                     lambda_seq = seq(1, 8, by = 0.25),
                                     alpha = 0.05,
                                     n_cores = 1,
                                     seed = 123)
# VBM with benchmarking
results <- run_sensitivity_analysis(weightlist=weightlist,
                                    Y="Y",
                                    treatment="treatment",
                                    data=nhanes.clean,
                                    approach="vbm",
                                    R2_seq = seq(0, 0.8, by = 0.01),
                                    alpha = 0.05,
                                    benchmarking = TRUE,
                                    n_bootstrap = 1000,
                                    n_cores = 1,
                                    seed = 331)
# VBM with correlation
results <- run_sensitivity_analysis(weightlist=weightlist,
                                    Y="Y",
                                    treatment="treatment",
                                    data=nhanes.clean,
                                    approach="vbm_w_cor",
                                    R2_seq = seq(0, 0.8, by=0.01),
                                    cor_eps_seq = rep(0.9, 81),
```

```
                                    n_cores = 1,
                                    alpha = 0.05)
# if only weights are proviced
only_weights <- run_sensitivity_analysis(weights=weightlist$weights,
                                          Y="Y",
                                          treatment="treatment",
                                          data=nhanes.clean,
                                          approach="msm",
                                          R2_seq = seq(1, 8, by = 0.25),
                                          alpha = 0.05,
                                          n_cores = 1,
                                          seed = 123)
# if weightit call is glm
glm_call <- quote(glm(formula = treatment ~ gender + age + income + income.missing
                      + education + smoking.now + smoking.ever + race,
                      data = nhanes.clean,family = binomial()))
glm_results <- run_sensitivity_analysis(weightlist=glm_call,
                                       Y="Y",
                                       treatment="treatment",
                                       data=nhanes.clean,
                                       approach="msm",
                                       R2_seq = seq(1, 8, by = 0.25),
                                       alpha = 0.05,
                                       seed = 123,
                                       n_cores = 1,
                                       benchmarking = TRUE)
```

Index